Universidade Estadual de Campinas
Instituto de Computação

Guilherme Henrique Santos Miranda

Ordenação de Permutações por Operações de Tamanho Limitado

CAMPINAS
2019

Guilherme Henrique Santos Miranda

Ordenação de Permutações por Operações de Tamanho Limitado

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Zanoni Dias
Coorientadora: Profa. Dra. Carla Negri Lintzmayer

Este exemplar corresponde à versão final da Dissertação defendida por Guilherme Henrique Santos Miranda e orientada pelo Prof. Dr. Zanoni Dias.

CAMPINAS
2019

Agência(s) de fomento e nº(s) de processo(s): CAPES
ORCID: https://orcid.org/0000-0001-5643-4527

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

M672o

Miranda, Guilherme Henrique Santos, 1995-
Ordenação de permutações por operações de tamanho limitado / Guilherme Henrique Santos Miranda. – Campinas, SP : [s.n.], 2019.

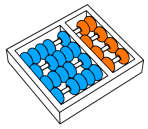
Orientador: Zanoni Dias.
Coorientador: Carla Negri Lintzmayer.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Biologia computacional. 2. Permutações (Matemática). 3. Rearranjo de genomas. 4. Algoritmos de aproximação. 5. Teoria da computação. 6. Ordenação (Computadores). I. Dias, Zanoni, 1975-. II. Lintzmayer, Carla Negri, 1990-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Sorting permutations by limited-size operations
Palavras-chave em inglês:
Computational biology
Permutations (Mathematics)
Genome rearrangements
Approximation algorithms
Theory of computing
Sorting (Electronic computers)
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Zanoni Dias [Orientador]
Pedro Henrique Del Bianco Hokama
Orlando Lee
Data de defesa: 21-02-2019
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação



Guilherme Henrique Santos Miranda

Ordenação de Permutações por Operações de Tamanho Limitado

Banca Examinadora:

- Prof. Dr. Zanoni Dias
  Universidade Estadual de Campinas
- Prof. Dr. Pedro Henrique Del Bianco Hokama
  Universidade Federal de Itajubá
- Prof. Dr. Orlando Lee
  Universidade Estadual de Campinas

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 22 de fevereiro de 2019

Agradecimentos

Pois é, no fim das contas consegui ser aprovado no processo seletivo de alguns programas de mestrado em universidades públicas que sempre sonhei em estudar. Há pouco mais de 2 anos atrás eu nunca imaginaria que as portas da Unicamp se abririam para mim, e que seria nela que eu transgrediria do título de bacharel para mestre. Sabe, apesar de ter muito orgulho da FIPP, a faculdade privada da onde vim, histórias como essa não são muito comuns por lá, e, assim, eu acabava por muitas vezes um pouco desacreditado de que era possível ter um lugarzinho para mim aqui.

A minha mãe, Ivonete, com certeza é a pessoa mais guerreira que já conheci. Dedicou anos de sua vida trabalhando dia e noite no Japão, sozinha, para que eu e meu irmão mais novo tivéssemos uma vida boa aqui no Brasil, enquanto a esperávamos voltar. Quando decidi fazer mestrado, você não mediu esforços em me apoiar. Fez horas extras e adiou seus planos para que tudo isso fosse possível sem que eu passasse nenhuma necessidade no meu primeiro semestre em que ainda não havia sido contemplado com uma bolsa de estudos. Mãe, nada disso seria possível sem você. Muitíssimo obrigado, por tudo!

Meu irmão, Leonardo, sempre foi o melhor amigo e a família mais presente que me acompanhou a maior parte da vida. Muitas vezes você tirou do pouco dinheirinho que ganha para me ajudar, e sempre fez isso sem exitar. Você é de mais, Léo. Muito obrigado!

Minha tia Dalva é a pessoa responsável por eu não mudar de ideia em escolher o curso de Ciência da Computação e é a principal pessoa responsável pela minha criação, junto ao meu tio Pedro e a minha avó, Marina. Tia, tio e vó, vocês sempre se importaram comigo e fizeram o melhor de vocês para me tornar uma boa pessoa. E sempre me motivaram a estudar. Eu agradeço vocês imensamente. Muito obrigado. Infelizmente não há espaço para citar um a um todos os familiares que me ajudaram, ou ajudaram a minha mãe, nessa trajetória. Aqui fica o meu agradecimento a todos vocês.

Por acreditar e confiar em mim, por toda sua empatia, empenho e presença, eu agradeço a você Zanoni. Não me vejo tendo mais sorte nessa trajetória do que ter tido você como orientador. Muito obrigado!

À Carla, minha orientadora, que além de excepcional no que faz, sempre se esforçou em, pacientemente, compartilhar um pouco do que sabe comigo. Muito obrigado por me ouvir, por me aconselhar e por a todo momento estar disponível para me ajudar.

Meus anos em Barão Geraldo com certeza foram mais felizes e inesquecíveis por conta das pessoas que neles, com amor, parceria e empatia, me acompanharam. Por terem sido incríveis comigo, e vocês sabem quem são, meus mais profundos agradecimentos!

Também agradeço à CAPES, por todo o indispensável fomento financeiro que foi disponibilizado a mim, e a todos do LOCo, pela companhia e companheirismo nesses anos. Ademais, ficam aqui meus agradecimentos ao SAE da Unicamp, pela ajuda financeira, que custeou parte da minha moradia e alimentação durante a minha estadia em Campinas.

Deus, o senhor é incrível. Obrigado pelas oportunidades e pessoas que me colocou nessa jornada e por cuidar das pessoas que amo. A minha história está apenas começando.

Resumo

Estimar a distância evolucionária entre genomas de dois organismos é uma tarefa desafiadora para Biologia Computacional. Uma das formas mais bem aceitas de estimar essa distância é considerar o tamanho da menor sequência de eventos de rearranjos necessários para transformar um genoma em outro, o que caracteriza o problema de distância de rearranjo. Diversos eventos de rearranjos podem ser considerados e, dentre os mais estudados, estão os rearranjos de reversão, que revertem um trecho de um genoma, e os de transposição, que trocam as posições de dois trechos adjacentes de um genoma. Computacionalmente, genomas podem ser representados como permutações de números inteiros e, com isso, o problema pode ser reduzido ao problema de calcular o número mínimo de operações (rearranjos) necessárias para transformar uma permutação na outra.

Dado um valor inteiro λ, uma operação é chamada de λ-operação se a mesma envolve no máximo λ elementos da permutação e uma permutação é chamada de λ-permutação se todos seus elementos estão a menos do que λ posições de distância de suas posições corretas na permutação ordenada. Desta forma, para os problemas de ordenação de permutações com e/ou sem sinais por reversões, por transposições, e por reversões e transposições, consideramos duas novas variações: (i) o problema de Ordenação de Permutações por λ-Operações, e (ii) o problema de Ordenação de λ-Permutações por λ-Operações, em que dada uma λ-permutação como entrada, o objetivo é ordená-la apenas utilizando λ-operações que geram λ-permutações como resultado.

Para (i), esta dissertação apresenta algoritmos com fatores de aproximação baseados no tamanho da permutação e/ou em λ, que estão divididos entre os que são melhores para valores grandes e pequenos de λ.

Para (ii), apresentamos limitantes inferiores e superiores sobre o número de λ-permutações existentes, algoritmos com fatores de aproximação $O(\lambda^2)$, $O(\lambda)$ e $O(1)$, e resultados sobre os diâmetros de ordenação. Além disso, um algoritmo com fator de aproximação $O(1)$ também é apresentado para o problema (i) considerando apenas λ-permutações dadas como entrada.

Para ambas as classes de problemas, são apresentados resultados experimentais que comparam como os algoritmos se comportam numa perspectiva prática.

Abstract

Estimating the evolutionary distance between genomes of two different organisms is a challenging task for Computational Biology. A well-accepted way to do this is by considering the size of the smallest sequence of rearrangement events needed to transform a genome into another, characterizing the rearrangement distance problem. Different rearrangement events can be considered and, among the most studied of them are the reversals, which revert a segment of a genome, and the transpositions, which swap two adjacent segments of a genome. Computationally, genomes can be represented as permutations of integers and, with this, the problem is reducible to the problem of calculating the minimum number of operations (rearrangements) needed to transform a permutation into another.

Given an integer λ, an operation is called λ-operation if it involves at most λ elements of a permutation and a permutation is called λ-permutation if it has all elements less than λ positions away from their correct positions in the sorted permutation. In this way, for the problems of sorting signed and/or unsigned permutations by reversals, by transpositions, and by reversals and transpositions, we are considering two new variations: (i) the problem of Sorting Permutations by λ-Operations, and (ii) the problem of Sorting λ-Permutations by λ-Operations, where, in this problem, the λ-operations are restricted to the ones which generate λ-permutations as a result.

For (i), this dissertation presents algorithms with approximation factors based on the size of the permutation and/or on λ, which are divided among the ones that are better for large and small values of λ.

For (ii), this dissertation presents lower bounds and upper bounds on the number of existent λ-permutations, algorithms with approximation factors $O(\lambda^2)$, $O(\lambda)$, and $O(1)$, and also results on the diameters. Moreover, an algorithm with approximation factor $O(1)$ is also shown for (i) considering only λ-permutations given as input.

Besides that, for both (i) and (ii), we present experimental results that compare how such algorithms work from a practical perspective.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Entender o processo evolutivo dos organismos é algo que sempre despertou curiosidade na humanidade e é um desafio importante para Biologia Computacional. A distância evolucionária entre genomas de dois organismos pode ser estimada com o tamanho da menor sequência de eventos de rearranjos necessários para transformar um genoma em outro, que representa o cenário mais provável de evolução, com base no *Princípio da Máxima Parcimônia*. Um rearranjo de genoma é um evento que ocorre com relativa raridade e modifica grandes trechos de um genoma, diferentemente das mutações pontuais, que afetam apenas moléculas individuais. Por possuir tais características, a estimativa da distância evolucionária com base nos eventos de rearranjos tende a ser mais precisa.

Computacionalmente, um genoma pode ser representado como uma permutação de números inteiros, assumindo que não existam genes duplicados e que ele seja composto por um único cromossomo linear. Uma permutação é dita com sinais (sem sinais) caso a orientação dos genes seja conhecida (desconhecida). Devido a tal representação do genoma, o problema de estimar a distância evolucionária com a quantidade mínima de eventos de rearranjo necessários para transformar um genoma em outro pode ser reduzido ao problema de calcular o número mínimo de operações necessárias para transformar uma permutação em outra. Além disso, podemos representar um dos genomas como a permutação identidade e, dessa forma, este problema é equivalente ao problema de encontrar a quantidade mínima de rearranjos necessários para ordenar uma permutação.

Na literatura foram consideradas diferentes operações de rearranjo, como por exemplo operações de reversões, que invertem um determinado segmento do genoma, e operações de transposições, que trocam de posição dois segmentos adjacentes na permutação. Estas operações podem possuir restrições extras, como sobre em quais partes do genoma serão aplicadas ou sobre a quantidade máxima de elementos do segmento afetado, sendo tal quantidade definida como tamanho da operação. O conjunto de eventos permitidos para transformar um genoma é dado por um *Modelo de Rearranjo*. Inicialmente, foram estudados modelos que permitiam apenas um tipo de evento de rearranjo [3, 5]. Entretanto, modelos que permitem mais do que um único tipo de evento foram estudados recentemente [27, 43, 49]. Esta dissertação estuda o problema de ordenação de permutações por operações de rearranjo com tamanhos limitados, utilizando modelos de rearranjo que permitem apenas operações de reversão e/ou apenas operações de transposição.

A relevância biológica para a restrição no tamanho do genoma parte da observação

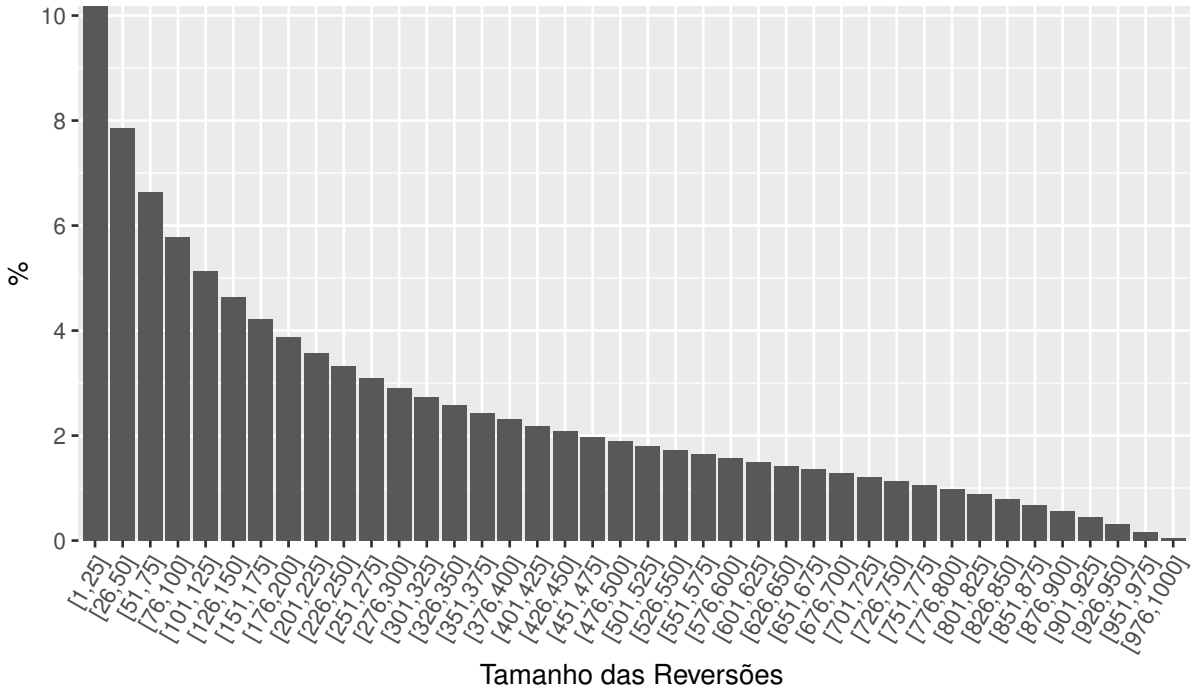


Figura 1.1: Frequência dos tamanhos das reversões dadas pelo *GRIMM* para o problema de Ordenação de Permutações com Sinais por Reversões, considerando 10000 permutações aleatórias de tamanho 1000.

de que eventos de rearranjo que afetam grandes trechos do genoma ocorrem com pouca frequência, prevalecendo rearranjos que envolvem um baixo número de genes [17, 37]. As figuras 1.1 e 1.2 mostram a frequência dos tamanhos de reversões utilizadas pela solução ótima, dada pelo *software GRIMM (Genome Rearrangements In Man and Mouse)* [47], para o problema de Ordenação de Permutações com Sinais por Reversões, considerando 10000 permutações arbitrárias de tamanho 1000. A Figura 1.1 mostra a solução regular do *GRIMM*, enquanto a Figura 1.2 mostra a solução quando solicitamos que reversões menores fossem priorizadas.

Para modelos que permitem apenas reversões ou apenas transposições de tamanho no máximo 2, o problema foi provado possuir solução em tempo polinomial [32]. Quando o tamanho máximo permitido é 3, os melhores resultados conhecidos são algoritmos de 2-aproximação [31] e $\frac{4}{3}$-aproximação [29, 30], para modelos considerando apenas reversões e apenas transposições, respectivamente.

Quando o tamanho limite das operações é dado por um valor λ arbitrário, as operações de reversões e transposições são chamadas de λ-reversões e λ-transposições, respectivamente.

Além disso, quando se impõe limite no tamanho das operações, faz sentido que os elementos da permutação origem não estejam muito distantes de suas posições corretas (em relação a permutação identidade). Dessa forma, nós chamamos de λ-permutação uma permutação na qual todos elementos estão a uma distância menor do que λ de suas respectivas posições na permutação identidade.

Nesta dissertação nós apresentamos resultados para os problemas de Ordenação de

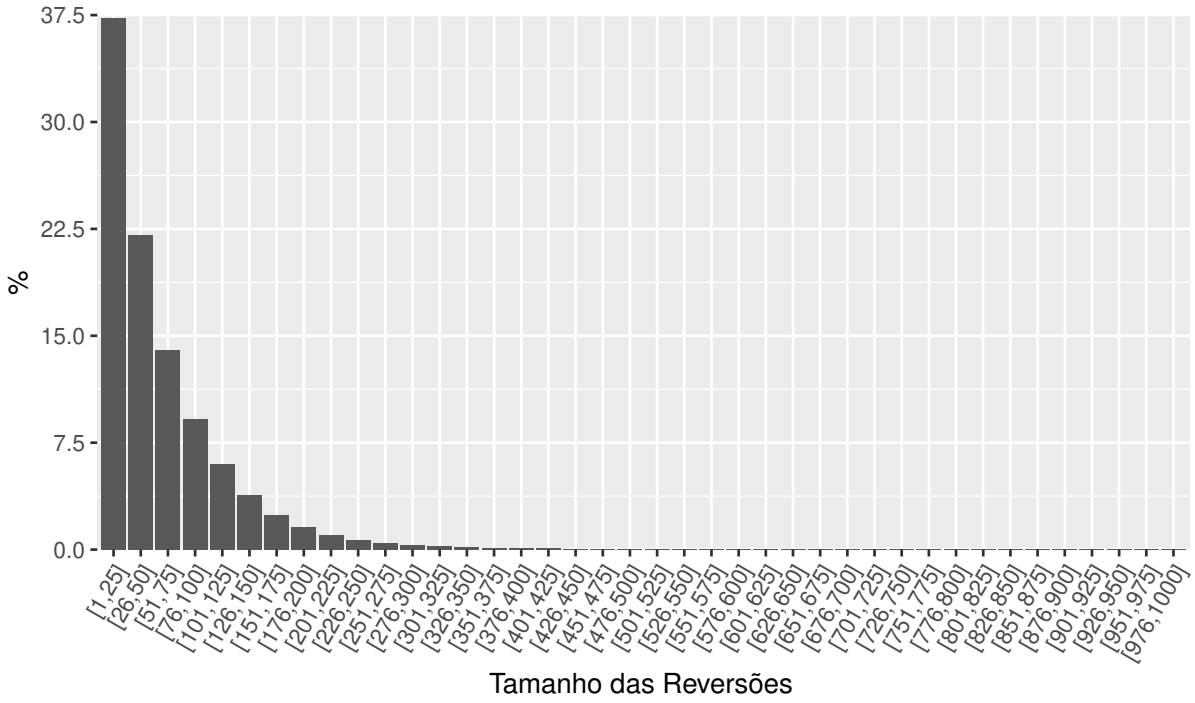


Figura 1.2: Frequência dos tamanhos das reversões dadas pelo *GRIMM* para o problema de Ordenação de Permutações com Sinais por Reversões quando operações pequenas foram priorizadas, considerando 10000 permutações aleatórias de tamanho 1000.

Permutações com e sem Sinais por λ-Reversões e/ou por λ-Transposições e também para os problemas de Ordenação de λ-Permutações com e sem Sinais por λ-Reversões e/ou por λ-Transposições. Os fatores dos algoritmos de aproximação que desenvolvemos podem ser visualizados resumidamente nas tabelas 1.1 e 1.2.

O restante desta dissertação é apresentado da seguinte forma. No Capítulo 2 são apresentadas as principais definições e conceitos necessários para o entendimento do texto. A revisão bibliográfica é apresentada no Capítulo 3. Os resultados obtidos para os problemas de Ordenação de Permutações por λ-Operações são apresentados no Capítulo 4. De forma semelhante, os resultados obtidos para os problemas de Ordenação de λ-Permutações por λ-Operações são apresentados no Capítulo 5. Finalmente, o Capítulo 6 apresenta a conclusão do trabalho.

Permutações	**Modelo de Rearranjo**	**Fator de Aproximação**	**Teoremas**
Com Sinais	λ-Reversões	$O((\frac{n}{\lambda})^2)$ e $O(\lambda^2)$	1 e 10
Sem Sinais		$O((\frac{n}{\lambda})^2)$ e $O(\lambda^2)$	2 e 9
Sem Sinais	λ-Transposições	$O((\frac{n}{\lambda})^2)$ e $O(\lambda^2)$	3 e 9
Com Sinais	λ-Reversões e λ-Transposições	$O((\frac{n}{\lambda})^2)$ e $O(\lambda^2)$	5 e 10
Sem Sinais		$O(\alpha(\frac{n}{\lambda})^2)$ e $O(\lambda^2)$	4 e 9

Tabela 1.1: Resultados apresentados para problemas de Ordenação de Permutações por λ-Operações. No problema de Ordenação de Permutações sem Sinais por λ-Reversões e λ-Transposições, α é o fator de aproximação do algoritmo de decomposição de ciclos para grafos de *breakpoints*.

λ-Permutações	**Modelo de Rearranjo**	**Fator de Aproximação**	**Teoremas (resp.)**
Com Sinais	λ-Reversões	$O(\lambda^2)$ e $O(\lambda)$	12 e 13
Sem Sinais		$O(\lambda^2)$ e $O(\lambda)$	11 e 13
Sem Sinais	λ-Transposições	$O(\lambda^2)$ e $O(1)$	11 e 14
Com Sinais	λ-Reversões e λ-Transposições	$O(\lambda^2)$ e $O(1)$	12 e 15
Sem Sinais		$O(\lambda^2)$ e $O(1)$	11 e 15

Tabela 1.2: Resultados apresentados para problemas de Ordenação de λ-Permutações por λ-Operações.

Capítulo 2

Fundamentação Teórica

Computacionalmente, um genoma é representado como uma n-upla que, ao assumirmos que não existe repetições de genes, pode ser dado como uma permutação $\pi = (\pi_1 \pi_2 \dots \pi_n)$ onde $\pi_i \in \{-n, -(n-1), \dots, -2, -1, +1, +2, \dots, +(n-1), +n\}$ e $|\pi_i| \neq |\pi_j| \iff i \neq j$. Os elementos π_i possuem sinal quando a orientação dos genes é conhecida e, nesse caso, a permutação é chamada de *permutação com sinais*. Quando não se tem conhecimento sobre a orientação dos genes, os sinais são omitidos e a permutação é denominada *permutação sem sinais*. Uma *permutação estendida* pode ser obtida ao se adicionar os elementos $\pi_0 = +0$ e $\pi_{n+1} = +(n+1)$ em π. Por convenção, nesta dissertação sempre iremos nos referir a permutações estendidas, entretanto o termo *estendida* e os elementos π_0 e π_{n+1} serão omitidos.

Dado um inteiro λ, dizemos que uma permutação π é uma λ-permutação se temos $||\pi_i| - i| < \lambda$ para todo $1 \leq i \leq n$.

A permutação inversa de π é denotada por π^{-1}. Esta permutação é tal que $\pi^{-1}_{|\pi_i|} = i$, para todo $1 \leq i \leq n$. Note que π_i^{-1} é a posição do elemento i (ou $-i$) em π. Por exemplo, dado $\pi = (-4\ +6\ -3\ -5\ +2\ +1)$, temos que $\pi^{-1} = (6\ 5\ 3\ 1\ 4\ 2)$.

Dadas duas permutações π e σ de tamanho n, a operação de composição "$\cdot$" gera uma nova permutação $\gamma = \pi \cdot \sigma = (\gamma_1\ \gamma_2\ \dots\ \gamma_n)$ onde $\gamma_i = -\pi_{|\sigma_i|}$ se $\sigma_i < 0$, ou $\gamma_i = \pi_{\sigma_i}$ caso contrário. Composições são utilizadas para indicar a aplicação de um rearranjo sobre uma permutação, como veremos a seguir.

Genomas podem ser representados como permutações e, assim, o problema de calcular a distância de rearranjo, dada pelo tamanho da menor sequência de operações necessárias para transformar um genoma em outro, pode ser modelado como o problema de encontrar o menor número de operações para ordenar uma permutação, ao representarmos um dos genomas como a permutação identidade $\iota = (1\ 2\ \dots\ n)$. Por esse motivo, tal problema é chamado de Problema de Ordenação de Permutações por Rearranjos.

Assim, dado um modelo de rearranjo M que determina as operações de rearranjo permitidas, o problema de ordenação de permutações por rearranjos consiste em encontrar a menor sequência de operações $\sigma^1, \sigma^2, \dots, \sigma^t$ pertencentes ao modelo tais que $\pi \cdot \sigma^1 \cdot \sigma^2 \cdot \dots \cdot \sigma^t = \iota$. O tamanho dessa sequência é definido como a distância de ordenação $d_M(\pi, \iota) = d_M(\pi) = t$.

2.1 Algoritmos de Aproximação

Seja I uma instância de um problema de minimização P. Seja OPT(I) o custo da solução ótima de I. Seja ALG(I) o custo da solução de I devolvida por um algoritmo ALG. Assim, ALG é um algoritmo de α-aproximação para P se possui complexidade de tempo polinomial e para toda instância I de P vale que ALG(I) $\leq \alpha$OPT(I), onde $\alpha \geq 1$. Por exemplo, seja I $= \{\pi\}$ uma instância para o problema de Ordenação de Permutações por Reversões, que contém uma permutação π. Um algoritmo de 2-aproximação para esse problema garante, em tempo polinomial, uma solução no máximo 2 vezes a distância de ordenação de π, para qualquer instância (ou permutação, nesse caso) dada como entrada. Além disso, como o valor da distância para os problemas apresentados são desconhecidos, iremos comparar a solução dos algoritmos apresentados com limitantes inferiores para as distâncias deles.

2.2 Reversões

Uma operação ou evento de reversão afeta um segmento do genoma e reverte seus elementos e, caso aplicada sobre uma permutação com sinais, cada elemento afetado tem seu sinal invertido. Formalmente, uma reversão $\rho(i, j)$, com $1 \leq i \leq j \leq n$, aplicada sobre uma permutação com sinais, é uma operação tal que:

$$\pi \cdot \rho(i,j) = (\pi_1\, \pi_2 \ldots \pi_{i-1}\, \underline{-\pi_j\, -\pi_{j-1}\, \ldots\, -\pi_{i+1}\, -\pi_i}\, \pi_{j+1} \ldots \pi_{n-1}\, \pi_n).$$

O tamanho de uma reversão é dado por $j - i + 1$. Por exemplo, dada a permutação com sinais $\pi = (+1\ +2\ -3\ +4\ +5)$, então $\pi \cdot \rho(2,4) = (+1\ \underline{-4\ +3\ -2}\ +5)$ e $\rho(2,4)$ é uma reversão de tamanho $4 - 2 + 1 = 3$. Quando uma reversão é aplicada em uma permutação sem sinais, o resultado é similar, mas não há mudanças de sinais nos elementos e também não consideramos reversões tais que $i = j$. Assim, dada a permutação sem sinais $\pi = (1\ 2\ 3\ 4\ 5)$, temos $\pi \cdot \rho(2,4) = (1\ \underline{4\ 3\ 2}\ 5)$.

As distâncias de ordenação por reversões são denotadas por $d_r(\pi)$ e $d_{\bar{r}}(\pi)$, respectivamente para permutações sem sinais e com sinais. A maior distância de ordenação por reversões entre todas as permutações de tamanho n é definida como o diâmetro $D_r(n)$, para permutações sem sinais, e $D_{\bar{r}}(n)$, para permutações com sinais. Quando o modelo é restrito apenas a operações de reversões, definimos o problema de encontrar a distância de rearranjo como o problema de Ordenação de Permutações por Reversões.

2.3 Transposições

Uma operação ou evento de transposição afeta dois segmentos adjacentes, trocando-os de lugar na permutação. Formalmente, uma transposição $\tau(i, j, k)$, para $1 \leq i < j < k \leq n + 1$, é um evento tal que:

$$\pi \cdot \tau(i,j,k) = (\pi_1\, \pi_2\, \ldots \pi_{i-1}\, \underline{\pi_j\, \ldots\, \pi_{k-1}}\, \underline{\pi_i \ldots\, \pi_{j-1}}\, \pi_k \ldots \pi_n).$$

O tamanho de uma transposição é dado por $k - i$. Por exemplo, se $\pi = (1\,2\,3\,4\,5)$, então $\pi \cdot \tau(1,3,5) = (\underline{3\,4}\,\underline{1\,2}\,5)$ e $\tau(1,3,5)$ é uma transposição de tamanho $5 - 1 = 4$. Uma vez que as operações de transposições não alteram os sinais dos elementos, não há diferença quando aplicadas em permutações com sinais.

A distância de ordenação por transposições é denotada por $d_t(\pi)$. O diâmetro $D_t(n)$ é definido como a maior distância de ordenação por transposições para todas as permutações de tamanho n. Quando o modelo é restrito apenas a operações de transposições, definimos o problema de encontrar a distância de rearranjo como o problema de Ordenação de Permutações por Transposições.

2.4 Transposições e Reversões

Quando o modelo de rearranjo permite ambas operações de reversão e transposição, o problema é definido como o problema de Ordenação de Permutações por Reversões e Transposições.

As distâncias de ordenação por reversões e transposições são denotadas por $d_{rt}(\pi)$ e $d_{\bar{r}t}(\pi)$, respectivamente para permutações sem sinais e com sinais. A maior distância de ordenação por reversões e transposições entre todas as permutações de tamanho n é definida como o diâmetro $D_{rt}(n)$ e $D_{\bar{r}t}(n)$, para permutações sem sinais e com sinais, respectivamente. É importante ressaltar que quando as duas operações são permitidas, temos $d_{rt}(\pi) \leq d_r(\pi)$, $d_{\bar{r}t}(\pi) \leq d_{\bar{r}}(\pi)$ e $d_{rt}(\pi) \leq d_t(\pi)$. Por exemplo, para $\pi = (1\,3\,4\,2\,7\,6\,5)$ temos: (i) $\pi \cdot \tau(2,4,5) \cdot \tau(5,7,8) \cdot \tau(6,7,8) = \iota$, com $d_t(\pi) = 3$; (ii) $\pi \cdot \rho(2,4) \cdot \rho(5,7) \cdot \rho(3,4) = \iota$, com $d_r(\pi) = 3$; e (iii) $\pi \cdot \tau(2,4,5) \cdot \rho(5,7) = \iota$, com $d_{rt}(\pi) = 2$.

2.5 λ-Operações e λ-Permutações

Chamaremos as operações de reversão e transposição de λ-reversão e λ-transposição, respectivamente, quando tais operações possuírem tamanho máximo λ. Para valores de $\lambda = 2$ e $\lambda = 3$, tais operações também são chamadas de super curtas e curtas, respectivamente. Além disso, as distâncias de ordenação de permutações sem sinais por λ-reversões, por λ-transposições e por ambas as operações serão denotadas por $d_r^\lambda(\pi)$, $d_t^\lambda(\pi)$ e $d_{rt}^\lambda(\pi)$, respectivamente. Para permutações com sinais, a distância de ordenação por λ-reversões é denotada por $d_{\bar{r}}^\lambda(\pi)$ e, quando ambas operações são permitidas, a distância de ordenação é denotada por $d_{\bar{r}t}^\lambda(\pi)$. Definiremos também os diâmetros $D_r^\lambda(n)$, $D_t^\lambda(n)$, $D_{rt}^\lambda(n)$, $D_{\bar{r}}^\lambda(n)$ e $D_{\bar{r}t}^\lambda(n)$ como o maior valor de $d_r^\lambda(\pi)$, $d_t^\lambda(\pi)$, $d_{rt}^\lambda(\pi)$, $d_{\bar{r}}^\lambda(\pi)$ e $d_{\bar{r}t}^\lambda(\pi)$, respectivamente, para todas as permutações π de tamanho n. Analogamente, as distâncias e diâmetros dos problemas de ordenação de λ-permutações por λ-operações serão denotadas por $d_\beta^{\lambda'}(\pi)$ e $D_\beta^{\lambda'}(\pi)$, para $\beta \in \{r, \bar{r}, t, rt, \bar{r}t\}$. Por exemplo, $d_r^2(\pi)$ é a distância de ordenação de permutações por reversões super curtas para uma permutação sem sinais π, enquanto $D_{\bar{r}t}^{3'}(n)$ é o diâmetro de ordenação de 3-permutações por reversões e transposições curtas para todas as 3-permutações com sinais de tamanho n.

2.6 Breakpoints

Para os problemas de ordenação de permutações sem sinais por reversões, e por reversões e transposições, um *breakpoint* é definido como um par de elementos (π_i, π_{i+1}), onde $0 \leq i \leq n$, tal que $|\pi_{i+1} - \pi_i| \neq 1$. Por exemplo, considerando a permutação $\pi =$ (1 3 4 2 7 6 5) dada como entrada para algum desses dois problemas, temos os *breakpoints* $(1,3), (4,2), (2,7)$ e $(5,8)$.

Para o problema de ordenação de permutações por transposições e para os problemas de ordenação de permutações com sinais por reversões e por ambas as operações, um *breakpoint* é definido como um par de elementos (π_i, π_{i+1}) tal que $\pi_{i+1} - \pi_i \neq 1$. Por exemplo, considerando a permutação $\pi = (+2\ -1\ +3\ +4\ +6\ +5)$ como entrada para algum desses três problemas, temos os *breakpoints* $(0,+2), (+2,-1), (-1,+3), (+4,+6)$, $(+6,+5)$ e $(+5,+7)$.

Em qualquer um dos problemas abordados, o número de *breakpoints* é denotado por $b(\pi)$. Assim, no primeiro exemplo do parágrafo anterior temos $b(\pi) = 4$ e, no segundo exemplo, temos $b(\pi) = 6$. As mesmas definições serão utilizadas para a versão dos problemas considerando λ-operações. Além disso, note que a única permutação com $b(\pi) = 0$ (em qualquer um dos problemas) é a permutação identidade.

O Lema 1 apresenta limitantes inferiores para os problemas de ordenação de permutações e λ-permutações abordados nesta dissertação.

Lema 1. *Para todas permutações (com ou sem sinais)* $\pi \neq \iota$ *e* $\lambda \geq 2$*, temos* $d_{\beta}^{\lambda}(\pi) \geq \frac{b(\pi)}{2}$ *e* $d_{\beta'}^{\lambda}(\pi) \geq \frac{b(\pi)}{3}$*, para* $\beta \in \{r, \bar{r}\}$ *e* $\beta' \in \{t, rt, \bar{r}t\}$*.*

Demonstração. Diretamente da observação de que uma λ-reversão ou uma λ-transposição remove no máximo 2 ou 3 *breakpoints*, respectivamente. □

Corolário 1. *Para todas* λ*-permutações (com ou sem sinais)* $\pi \neq \iota$ *e* $\lambda \geq 2$*, temos* $d_{\beta}^{\lambda'}(\pi) \geq \frac{b(\pi)}{2}$ *e* $d_{\beta'}^{\lambda'}(\pi) \geq \frac{b(\pi)}{3}$*, para* $\beta \in \{r, \bar{r}\}$ *e* $\beta' \in \{t, rt, \bar{r}t\}$*.*

2.7 Strips

Uma *strip* é definida como uma sequência maximal $(\pi_i \ \ldots \ \pi_j)$ de elementos da permutação tal que não existem *breakpoints* entre quaisquer pares (π_k, π_{k+1}), onde $i \leq k \leq j-1$. Por exemplo, considerando o problema de Ordenação de Permutações (ou λ-Permutações) sem Sinais por Reversões, a permutação $\pi = (\underline{1}\ \underline{3\ 4}\ \underline{2}\ \underline{7\ 6\ 5})$ possui 4 *strips*, destacadas na permutação. Considerando o problema de Ordenação de Permutações (ou λ-Permutações) com Sinais por Reversões, a permutação $\pi = (\underline{+1\ +2}\ \underline{+5}\ \underline{+4}\ \underline{-3}\ \underline{+6})$ possui 5 *strips*, também destacadas na permutação.

Para os problemas de ordenação que envolvem permutações ou λ-permutações sem sinais, chamamos uma *strip* de crescente (resp. decrescente) caso os elementos da mesma estejam em ordem crescente (resp. decrescente). *Strips* que possuem um único elemento também são consideradas crescentes. Por exemplo, considerando o problema de ordenação de permutações sem sinais utilizando ambas as operações e $\pi =$ (6 4 5 3 2 1), temos duas *strip* crescentes (6) e (4 5), e uma *strip* decrescente (3 2 1). Observe, porém, que o

segmento (3 2 1) não é uma *strip* decrescente para o problema de Ordenação de λ-Permutações por λ-Transposições. Isto ocorre pois, diretamente da nossa definição de *strips* e *breakpoints*, não há *strips* decrescentes para este problema.

Para os problemas de ordenação que envolvem permutações ou λ-permutações com sinais, se os elementos de uma *strip* são positivos, então ela é chamada de crescente. Caso contrário, ela é chamada de decrescente. *Strips* que contém somente um elemento são consideradas crescentes se tal elemento é positivo e são consideradas decrescentes caso contrário. Por exemplo, considerando os problemas de ordenação de λ-permutações com sinais e $\pi = (+6\ -5\ -4\ +1\ +2\ -3)$, temos duas *strips* crescentes $(+6)$ e $(+1\ +2)$, e duas *strips* decrescentes $(-5\ -4)$ e (-3). Note que, apesar dos elementos da *strip* $(-5\ -4)$ estarem em ordem crescente, ela ainda é uma *strip* decrescente, de acordo com a nossa definição.

O número de elementos em uma *strip* S de uma λ-permutação π será denotado por $|S|$.

Lema 2. *Seja π uma λ-permutação e seja $|\pi_j| = i$ o menor elemento fora de lugar em π. A strip S que contém π_j é tal que $|S| \leq \lambda - 1$.*

Demonstração. Suponha primeiramente $S = (\pi_j \ \ldots \ \pi_k)$ como uma *strip* crescente. Seja $R = (\pi_i \ \ldots \ \pi_{j-1})$ o segmento que possui os elementos à esquerda de S. Note que o valor absoluto de qualquer elemento em R é maior que qualquer elemento em S. Assim, $|\pi_i| > |\pi_k| \geq k-1$. Por contradição, suponha que $|S| \geq \lambda$. Logo, temos $|S| = k-j+1 \geq \lambda$ e $j \leq k - j + 1$. Uma vez que $i < j$, segue que $i \leq k - \lambda < |\pi_i|$, e então $||\pi_i| - i| = |\pi_i| - i \geq k - (k - \lambda) = \lambda > \lambda - 1$, que é uma contradição à definição de λ-permutação.

A prova segue por simetria para quando π_j está em uma *strip* decrescente. □

2.8 Entropia

A entropia de um elemento π_i, denotada por $\text{ent}(\pi_i)$, é dada por $||\pi_i| - i|$, isto é, a distância entre π_i e sua posição correta em ι. A entropia de uma permutação π, denotada por $\text{ent}(\pi)$, é dada pela soma de todos os valores de $\text{ent}(\pi_i)$, para $1 \leq i \leq n$. Por exemplo, a entropia de uma permutação sem sinais $\pi = (2\ 3\ 5\ 4\ 1)$ é $\text{ent}(\pi) = 1+1+2+0+4 = 8$. Denotamos por $E_\pi^{even^-}$ (resp. $E_\pi^{odd^+}$) o conjunto de elementos negativos (resp. positivos) em π tais que $\text{ent}(\pi_i)$ é par (resp. ímpar). Como um exemplo, dada a permutação com sinais $\pi = (-5\ +2\ -1\ -3\ +4)$, temos $E_\pi^{even^-} = \{-5, -1\}$, pois $\text{ent}(-5) = 4$ e $\text{ent}(-1) = 2$, e temos $E_\pi^{odd^+} = \{+4\}$, pois $\text{ent}(+4) = 1$. Além disso, note que os elementos $+2$ e -3 não estão em nenhum dos dois conjuntos, pois $\text{ent}(+2) = 0$ e $\text{ent}(-3) = 1$.

2.9 Inversões

Para uma permutação (λ-permutação ou não) π com ou sem sinais, uma inversão é definida como um par de elementos (π_i, π_j) tal que $\pi_i < \pi_j$ e $1 \leq j < i \leq n$ e o número de inversões é denotado por $\text{Inv}(\pi)$. Por exemplo, para $\pi = (3\ 1\ 4\ 2)$, os pares (π_1, π_2), (π_1, π_4) e (π_3, π_4) são as inversões de π, logo $\text{Inv}(\pi) = 3$. O Lema 3 apresenta um limitante superior

no número de inversões que pode ser removido por uma λ-operação. Ele implica no Corolário 2, que mostra limitantes inferiores para os problemas de ordenação que estamos considerando. Ambos são utilizados pelos algoritmos apresentados nas seções 4.2.3 e 5.3.

Lema 3. *Seja π uma permutação sem sinais. Seja π' a permutação resultante após aplicarmos uma λ-operação em π. Então,* $\mathrm{Inv}(\pi) - \mathrm{Inv}(\pi') \leq \lambda(\lambda-1)/2$.

Demonstração. Diretamente da observação de que qualquer λ-reversão ou λ-transposição pode remover ou adicionar no máximo $\lambda(\lambda-1)/2$ inversões. □

Uma vez que a única permutação sem sinais com $\mathrm{Inv}(\pi) = 0$ é a identidade, obtemos limitantes inferiores para os problemas de ordenação de permutações e λ-permutações sem sinais no Corolário 2.

Corolário 2. *Para todas permutações sem sinais $\pi \neq \iota$ e $\lambda \geq 2$, temos*

$$d_\beta^\lambda(\pi) \geq \frac{\mathrm{Inv}(\pi)}{\lambda(\lambda-1)/2} \text{ e } d_\beta^{\lambda'}(\pi) \geq \frac{\mathrm{Inv}(\pi)}{\lambda(\lambda-1)/2},$$

para $\beta \in \{r, rt, t\}$.

2.10 Inversões e Entropia

Os lemas 4, 5, e 6 utilizam definições de inversões e entropia e são úteis para obter limitantes inferiores para os problemas de ordenação de permutações e de λ-permutações com sinais por λ-operações, apresentados nas seções 4.2.3 e 5.3.

Lema 4. *(Galvão et al. [23]) Seja π uma permutação com sinais e seja $\pi' = \pi \cdot \rho(i, i+1)$. Então, $|E_\pi^{even^-}| + |E_\pi^{odd^+}| = |E_{\pi'}^{even^-}| + |E_{\pi'}^{odd^+}|$.*

Lema 5. *Seja π uma permutação com sinais. Seja π' a permutação resultante depois que uma λ-operação é aplicada sobre π. Então, $(|E_\pi^{even^-}| + |E_\pi^{odd^+}|) - (|E_{\pi'}^{even^-}| + |E_{\pi'}^{odd^+}|) \leq \lambda$.*

Demonstração. Uma λ-operação envolve no máximo λ elementos. □

Seja $\mathrm{score}_{inv}(\pi, \sigma) = (\mathrm{Inv}(\pi) + |E_\pi^{even^-}| + |E_\pi^{odd^+}|) - (|\mathrm{Inv}(\pi \cdot \sigma)| + |E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|)$ a pontuação de inversões de uma λ-operação σ aplicada sobre uma permutação (ou uma λ-permutação) π.

Lema 6. *Para qualquer permutação π e λ-operação σ, temos* $\mathrm{score}_{inv}(\pi, \sigma) \leq \lambda\frac{(\lambda-1)}{2} + \lambda$.

Demonstração. Diretamente dos lemas 3 e 5. □

Já que a única permutação com sinais tal que $\mathrm{Inv}(\pi) = |E_\pi^{even^-}| = |E_\pi^{odd^+}| = 0$ é a identidade, obtemos os limitantes inferiores para os problemas de ordenação de permutações e λ-permutações com sinais por λ-operações no Corolário 3.

Corolário 3. *Para todas as permutações com sinais $\pi \neq \iota$ e todos $\lambda \geq 2$, temos*

$$d_\beta^\lambda(\pi) \geq 2\left(\frac{\mathrm{Inv}(\pi) + |E_\pi^{even^-}| + |E_\pi^{odd^+}|}{\lambda(\lambda-1)+2\lambda}\right) \text{ e } d_\beta^{\lambda'}(\pi) \geq 2\left(\frac{\mathrm{Inv}(\pi) + |E_\pi^{even^-}| + |E_\pi^{odd^+}|}{\lambda(\lambda-1)+2\lambda}\right),$$

para $\beta \in \{\bar{r}, \bar{r}t\}$.

Capítulo 3

Revisão Bibliográfica

Neste capítulo é apresentado o estado da arte para os problemas de ordenação de permutações por reversões, por transposições e por transposições e reversões, com diferentes restrições.

3.1 Ordenação de Permutações por Reversões

O problema de Ordenação de Permutações sem Sinais por Reversões foi provado estar na classe de problemas NP-Difícil [12] e diversos pesquisadores apresentaram algoritmos de aproximação para o problema. Um dos primeiros resultados foi obtido por Bafna e Pevzner [3] com um algoritmo de 1.75-aproximação. Posteriormente, foi proposto por Christie [15] um algoritmo com fator de aproximação 1.5. Berman e coautores [7] apresentaram um algoritmo de 1.375-aproximação, que é o melhor conhecido para o problema. Além disso, um limitante inferior para a distância, $d_r(\pi) \geq \frac{b_r(\pi)}{2}$, foi apresentado por Bafna e Pevzner [4] e um limite superior, $d_r(\pi) \leq b_r(\pi) - 1$, foi apresentado por Kececioglu e Sankoff [35].

Curiosamente, o problema de Ordenação de Permutações com Sinais por Reversões possui algoritmo exato polinomial. A primeira versão foi proposta por Hannenhalli e Pevzner [28], que posteriormente foi simplificada por Bergeron [6]. O melhor algoritmo proposto na literatura foi dado por Tannier e coautores [46], com complexidade de tempo sub-quadrática. Além disso, um algoritmo linear foi proposto por Bader e coautores [2] para quando se deseja apenas a distância de rearranjo.

3.2 Ordenação de Permutações por Transposições

O problema de Ordenação de Permutações por Transposições foi introduzido por Bafna e Pevzner [5] em 1995. A complexidade computacional desse problema permaneceu em aberto por mais de 15 anos, até que Bulteau e coatores [10] provaram que ele pertence a classe de problemas NP-Difícil.

O primeiro algoritmo para o problema foi proposto pelos mesmos autores que o introduziram, Bafna e Pevzner [5], com fator de aproximação 1.5 e complexidade de tempo

$O(n^2)$. Foi também proposto pelos autores limitantes inferior $d_t(\pi) \geq \frac{b_t(\pi)}{3}$ e superior $d_t(\pi) \leq b_t(\pi)$ para as distâncias de rearranjo.

Christie [16] propôs posteriormente um algoritmo com complexidade de tempo $O(n^4)$ mais simples de ser implementado, ainda com fator de aproximação 1.5. Um outro algoritmo com tempo quadrático e implementação mais simples foi proposto por Walter e coautores [50], entretanto com fator de aproximação 2.25. O melhor algoritmo conhecido para o problema possui fator de aproximação 1.375, complexidade de tempo $O(n^2)$ e foi proposto em 2006 por Elias e Hartman [20].

Rusu [44] apresentou uma maneira de implementar vários algoritmos da literatura para os problema de ordenação de permutações por transposições e/ou reversões em tempo $O(n \log n)$, com o uso de *log-lists*.

3.3 Ordenação de Permutações por Reversões e Transposições

Quando ambas as operações são permitidas no modelo de rearranjo, a complexidade dos problemas de ordenação de permutações com e sem sinais está em aberto.

Walter e coautores [49] apresentaram limites inferiores para a distância e diâmetro, além de algoritmos de 3-aproximação e 2-aproximação, para as versões sem sinais e com sinais, respectivamente.

Posteriormente, Rahman e coautores [43] apresentaram, em 2008, um algoritmo de 2α-aproximação para permutações sem sinais, sendo α o fator de aproximação do algoritmo para decomposição de ciclos de um grafo de *breakpoints*. Dado o melhor valor de k conhecido [14], o fator de aproximação deste algoritmo é de $2.8334 + \epsilon$, para $\epsilon > 0$.

3.4 Ordenação de Permutações por Operações de Prefixos e Sufixos

A versão do problema de ordenação de permutações por reversões com a restrição de que as operações podem ser aplicadas somente nos prefixos da permutação foi introduzida por Dweighter [19] com o nome de Problema da Ordenação de Panquecas, em 1975. A descrição desse problema consistia, resumidamente, em ordenar uma pilha de panquecas de acordo com seus respectivos tamanhos, de forma que a menor panqueca ficasse no topo, a segunda menor panqueca logo abaixo dela, e assim sucessivamente, até a base da pilha, que conteria a maior panqueca. A reinterpretação deste problema para um problema de Rearranjo de Genomas foi dada por Pevzner e Waterman [42], em 1995 e o problema foi provado ser NP-Difícil em 2011 por Bulteau e coautores [11]. Gates e Papadimitriou [26] estudaram o Problema da Ordenação de Panquecas em 1979, a fim de encontrar a maior quantidade de movimentos necessários para ordenar qualquer pilha de panquecas e mostraram que ela está entre $\frac{17n}{16}$ e $\frac{5n+5}{3}$. Quando o objetivo é encontrar a menor quantidade de movimentos para ordenar uma dada pilha de panquecas, o melhor resultado conhecido é um algoritmo de 2-aproximação [21].

O Problema de Ordenação por Transposições de Prefixo foi introduzido por Dias e Meidanis [18]. O melhor resultado conhecido para este problema é um algoritmo de 2-aproximação, dado pelos mesmos autores.

Sharmin e coautores [45] propuseram duas novas versões para o problema da ordenação de panquecas: (i) permitindo utilizar reversões e/ou transposições de prefixo e (ii) permitindo utilizar transposições, reversões e/ou transreversões de prefixo, onde uma transreversão é uma operação dada pela combinação de uma transposição e uma reversão. Estes mesmos autores apresentaram algoritmos de 3-aproximação e 2-aproximação para os problemas (i) e (ii), respectivamente.

Lintzmayer e coautores [39] introduziram problemas de ordenação de permutações (com e sem sinais) considerando operações tanto de sufixo quanto de prefixo, para reversões e/ou transposições. Os autores apresentaram algoritmos com fatores de aproximação 2 e $2 + O(1/b(\pi))$ para tais problemas.

3.5 Ordenação de Permutações por Operações Curtas e Super Curtas

Quando o tamanho da operação é limitado a no máximo 2, Jerrum [32] mostrou que o problema de ordenação de permutação sem sinais por reversões (ou transposições) super curtas possui solução em tempo polinomial, com a prova dada por Knuth [36].

Quando o tamanho é limitado a no máximo 3, Heath e Vergara apresentaram um algoritmo de $\frac{4}{3}$-aproximação para o problema de ordenação de permutação por transposições curtas [30] e posteriormente apresentaram um algoritmo com fator de aproximação 2 para o problema de ordenação de permutação sem sinais por reversões curtas [31], o melhor resultado conhecido. Além disso, Vergara [48] também mostrou que o algoritmo de $\frac{4}{3}$-aproximação para o problema de ordenação por transposições curtas é também um algoritmo de 2-aproximação para o problema de ordenação por reversões e transposições curtas. Em 2012, um algoritmo com fator de aproximação $(1+1/\mathrm{Inv}(\pi))$ para o problema de ordenação de permutações por transposições curtas foi dado por Jiang e coautores [34]. Dois anos depois, um outro algoritmo com fator de aproximação $\frac{5}{4}$ foi proposto por Jiang e coautores [33] também para o problema de ordenação por transposições curtas.

Três algoritmos de aproximação para o problema de ordenação de permutações com sinais por reversões curtas foram propostos por Galvão e Dias [24] em 2014, sendo o melhor deles com fator de aproximação 9. No ano seguinte, Galvão e coautores [23] apresentaram algoritmos exatos e com complexidade polinomial para os problemas de ordenação de permutações com sinais por reversões super curtas e ordenação de permutações com sinais por reversões e transposições super curtas, além de algoritmos com fator de aproximação 5 e 3 para os problemas de ordenação de permutações com sinais por reversões curtas e ordenação de permutações com sinais por reversões e transposições curtas, respectivamente. Além disso, também foi mostrado pelos autores que o fator de aproximação esperado para o algoritmo de ordenação de permutações por reversões curtas é 3, quando a permutação possui 12 ou mais elementos.

Para o problema de ordenação de permutações circulares, onde se considera que o

Permutações	Modelo de Rearranjo	Complexidade	Melhor Resultado
Com Sinais	Reversões	Polinomial [28]	Tempo $O(n^{\frac{3}{2}})$ [46]
Sem Sinais		NP-Difícil [12]	1.375-aproximação [7]
Sem Sinais	Transposições	NP-Difícil [10]	1.375-aproximação [20]
Com Sinais	Reversões e Transposições	?	2-aproximação [49]
Sem Sinais		?	2α-aproximação [43]
Com Sinais	Reversões Super Curtas	Polinomial [23]	Tempo $O(n^3)$ [23]
Sem Sinais		Polinomial [36, p. 108]	Tempo $O(n^2)$ [32]
Sem Sinais	Transposições Super Curtas	Polinomial [36, p. 108]	Tempo $O(n^2)$ [32]
Com Sinais	Reversões e Transposições Super Curtas	Polinomial [23]	Tempo $O(n^3)$ [23]
Sem Sinais		Polinomial [36, p. 108]	Tempo $O(n^2)$ [32]
Com Sinais	Reversões Curtas	?	5-aproximação [23]
Sem Sinais		?	2-aproximação [31]
Sem Sinais	Transposições Curtas	?	$\frac{5}{4}$-aproximação [33]
Com Sinais	Reversões e Transposições Curtas	?	3-aproximação [23]
Sem Sinais		?	2-aproximação [48]

Tabela 3.1: Estado da arte para problemas de ordenação de permutações.

último e o primeiro elemento da permutação são adjacentes, também existe a opção de utilizar apenas operações super curtas. Para esse problema, Galvão e coautores [22] apresentaram, em 2015, um algoritmo com tempo polinomial para permutações com sinais e, ainda, disponibilizaram no ano seguinte uma ferramenta *web* [25] para inferência filogenética baseada em rearranjos usando reversões super curtas.

3.6 Resumo: Estado da Arte

O estado da arte para as diferentes variações do problema de Ordenação de Permutações por Rearranjos é apresentado, resumidamente, na Tabela 3.1.

Capítulo 4

Ordenação de Permutações por λ-Operações

Neste capítulo são apresentados algoritmos de aproximação para os 5 problemas de Ordenação de Permutações por λ-Operações que estamos estudando. Os algoritmos apresentados neste capítulo são apresentados em duas seções. A Seção 4.1 apresenta algoritmos de aproximação que possuem melhores fatores de aproximação para valores grandes de λ, enquanto a Seção 4.2 apresenta algoritmos que são melhores para valores pequenos de λ. Finalmente, a Seção 4.3 apresenta resultados experimentais de tais algoritmos.

Além disso, ressaltamos que todos os algoritmos apresentados neste capítulo funcionam para quaisquer valores de $\lambda \geq 2$.

4.1 Algoritmos de Aproximação para Valores Grandes de λ

Os algoritmos de aproximação apresentados nesta seção foram obtidos a partir do uso de algoritmos já existentes na literatura para os problemas em que não temos a restrição no tamanho dos rearranjos.

Assim, o primeiro passo é relacionar a distância de cada problema envolvendo λ-operações com cada problema envolvendo as operações sem tal restrição. Isto pode ser visto no Lema 7.

Lema 7. *Para todas as permutações π e todos $\lambda \geq 2$, temos que $d_r^\lambda(\pi) \geq d_r(\pi)$, $d_{\overline{r}}^\lambda(\pi) \geq d_r(\pi)$, $d_t^\lambda(\pi) \geq d_t(\pi)$, $d_{rt}^\lambda(\pi) \geq d_{rt}(\pi)$, e $d_{\overline{r}t}^\lambda(\pi) \geq d_{\overline{r}t}(\pi)$.*

Demonstração. Qualquer sequência de ordenação em que o tamanho dos rearranjos é limitado por λ também é uma sequência válida para quando os tamanhos das operações não são limitados. □

Os lemas 8 e 9 mostram como simular qualquer reversão e transposição com uma sequência de λ-reversões e λ-transposições, respectivamente.

Lema 8. *Para uma permutação (com ou sem sinais) π e $\lambda \geq 2$, o efeito de uma reversão $\rho(i,j)$ de tamanho $j-i+1 > \lambda$ pode ser obtido por no máximo $\frac{q(q+1)}{2}$ λ-reversões, onde $q = \left\lceil \frac{j-i+1}{\lfloor \lambda/2 \rfloor} \right\rceil$.*

Demonstração. Inicialmente dividimos o segmento de π que contém os elementos da posição i até a posição j em subsegmentos de tamanho $\lfloor \lambda/2 \rfloor$, exceto talvez pelo mais próximo de j que pode ser menor, resultando em $q = \left\lceil \frac{j-i+1}{\lfloor \lambda/2 \rfloor} \right\rceil$ subsegmentos. Formalmente, para cada $1 \leq \ell < q$, o ℓ-ésimo subsegmento contém elementos de π da posição $i + \lfloor \lambda/2 \rfloor(\ell - 1)$ até $i + \lfloor \lambda/2 \rfloor \ell - 1$, e o q-ésimo subsegmento contém elementos de π da posição $i + \lfloor \lambda/2 \rfloor(q-1)$ até j. Daqui em diante os subsegmentos serão definidos pelos elementos contidos neles em π. Por exemplo, em π o q-ésimo subsegmento termina na posição j mas em $\pi \cdot \rho(i,j)$ ele começa na posição i. Mesmo assim, este segmento é referido como o q-ésimo subsegmento em $\pi \cdot \rho(i,j)$.

A ideia agora é mover cada subsegmento até suas respectivas posições em $\pi \cdot \rho(i,j)$ ao trocar um subsegmento com outro à sua direita. As reversões usadas terão tamanhos de no máximo $2\lfloor \lambda/2 \rfloor$, e, portanto, elas são λ-reversões.

Para cada valor de ℓ, de $\ell = 1$ até $\ell = q - 1$, aplicamos uma sequência de λ-reversões que primeiro troca o ℓ-ésimo subsegmento com o $(\ell+1)$-ésimo subsegmento, então troca o ℓ-ésimo subsegmento com o $(\ell+2)$-ésimo, e assim por diante, até a troca do ℓ-ésimo subsegmento com o q-ésimo subsegmento. Note que depois de aplicar esta sequência na etapa ℓ sobre o ℓ-ésimo subsegmento, o mesmo é colocado em sua posição final (em relação a $\pi \cdot \rho(i,j)$) e o $(\ell+1)$-ésimo subsegmento está atualmente começando na posição i. Também, depois de aplicar a sequência final (de λ-reversões) sobre o $(q-1)$-ésimo subsegmento (que é o último considerado), os subsegmentos $q-1$ e q são colocados em suas posições finais.

Note que exatamente $q - \ell$ λ-reversões são realizadas na etapa ℓ. Temos então que um total de $(q-1) + (q-2) + \cdots + 1 = \frac{q(q-1)}{2}$ λ-reversões são feitas para posicionar todos os subsegmentos. Agora, se q é par, então no final do processo teremos exatamente $\pi \cdot \rho(i,j)$. Caso contrário, todos os subsegmentos ainda terão que ser revertidos e, dessa forma, outras q λ-reversões de tamanho $\lfloor \lambda/2 \rfloor$ (exceto, talvez, pela última delas sobre o q-ésimo subsegmento) são aplicadas, uma para cada subsegmento. Assim, o efeito de uma reversão pode ser obtido com $\frac{q(q-1)}{2} + q = \frac{q(q+1)}{2}$ λ-reversões. □

Como um exemplo do lema anterior, seja $\pi = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$ uma permutação sem sinais e suponha que queremos obter $\pi \cdot \rho(2,6) = (1\ 6\ 5\ 4\ 3\ 2\ 7\ 8)$ com 4-reversões. Sejam $A = (2\ 3)$, $B = (4\ 5)$, e $C = (6)$ os subsegmentos que serão movidos, como descritos no Lema 8. Note que a soma dos tamanhos de quaisquer dois segmentos consecutivos é menor ou igual a $\lambda = 4$. O processo descrito no lema primeiramente troca A com B, gerando $(1\ 5\ 4\ 3\ 2\ 6\ 7\ 8)$, e então com C, gerando $(1\ 5\ 4\ 6\ 2\ 3\ 7\ 8)$. Então, troca B com C, gerando $(1\ 6\ 4\ 5\ 2\ 3\ 7\ 8)$, e o processo termina. Note que $q = \lceil (j-i+1)/\lfloor \lambda/2 \rfloor \rceil = \lceil (6-2+1)/\lfloor 4/2 \rfloor \rceil = \lceil 5/2 \rceil = 3$ é um número ímpar, então apesar dos segmentos estarem em suas posições corretas, ainda temos que reverter A e B (já que $|C| = 1$ e π é uma permutação sem sinais), pois os elementos contidos nesses segmentos estão em ordem reversa do esperado. Note também que, se π fosse uma permutação com sinais, os sinais

dos elementos também estariam o oposto do esperado neste caso. Assim, nós obtemos $\pi \cdot \rho(2,6)$ com um total de $5 = 3 + 2 \leq q(q+1)/2 = (3 \times 4)/2 = 6$ 4-reversões.

Lema 9. *Para uma permutação (com ou sem sinais) π e $\lambda \geq 2$, o efeito de uma transposição $\tau(i,j,k)$ de tamanho $k - i > \lambda$ pode ser obtido por no máximo $\left\lceil \frac{j-i}{\lceil \lambda/2 \rceil} \right\rceil \left\lceil \frac{k-j}{\lfloor \lambda/2 \rfloor} \right\rceil$ λ-transposições.*

Demonstração. Denote por P o primeiro segmento da transposição, que contém elementos de π compreendidos entre as posições i e $j-1$, e denote por S o segundo segmento, que contém elementos de π compreendidos entre as posições j e $k-1$. Nós dividimos P em $p = \lceil (j-i)/\lceil \lambda/2 \rceil \rceil$ subsegmentos de tamanho $\lceil \lambda/2 \rceil$, exceto talvez por aquele que termina em $j-1$, e dividimos S em $s = \lceil (k-j)/\lfloor \lambda/2 \rfloor \rceil$ subsegmentos de tamanho $\lfloor \lambda/2 \rfloor$, também exceto talvez por aquele que termina em $k-1$, na seguinte maneira. Para $1 \leq \ell < p$, o ℓ-ésimo subsegmento de P contém elementos de π da posição $i + \lceil \lambda/2 \rceil (\ell - 1)$ até $i + \lceil \lambda/2 \rceil \ell - 1$ e o p-ésimo subsegmento de P contém elementos de π da posição $i + \lceil \lambda/2 \rceil (p-1)$ até $j-1$. Para $1 \leq \ell < s$, o ℓ-ésimo subsegmento de S contém elementos de π das posições $j + \lfloor \lambda/2 \rfloor (\ell - 1)$ até $j + \lfloor \lambda/2 \rfloor \ell - 1$ e o s-ésimo subsegmento de S contém elementos de π da posição $j + \lfloor \lambda/2 \rfloor (s-1)$ até $k-1$. Como no Lema 8, os segmentos P e S e seus subsegmentos são definidos pelos elementos contidos neles.

A ideia agora é mover cada subsegmento de P até suas respectivas posições em $\pi \cdot \tau(i,j,k)$ trocando-os com subsegmentos de S. As transposições utilizadas terão tamanhos de no máximo $\lceil \lambda/2 \rceil + \lfloor \lambda/2 \rfloor$ e então elas são λ-transposições.

Para cada valor de ℓ, de $\ell = p$ até $\ell = 1$, nós aplicamos uma sequência de λ-transposições que primeiro troca o ℓ-ésimo subsegmento de P com o primeiro subsegmento de S, então troca o ℓ-ésimo subsegmento de P com o segundo subsegmento de S, e assim por diante, até a troca do ℓ-ésimo subsegmento de P com o s-ésimo subsegmento de S. Note que depois de aplicar a sequência na etapa ℓ, o ℓ-ésimo subsegmento de P estará em sua posição final (em relação a $\pi \cdot \tau(i,j,k)$) e o segmento S continua igual a como ele está em π, mas começando à direita do $(\ell-1)$-ésimo subsegmento P. Assim, a próxima iteração (para $\ell - 1$) irá trocar o $(\ell-1)$-ésimo subsegmento de P com todos subsegmentos de S e colocá-lo na sua posição final.

Note que exatamente s λ-transposições são realizadas sobre o ℓ-ésimo subsegmento de P, então um total de $ps = \lceil (j-i)/\lceil \lambda/2 \rceil \rceil \lceil (k-j)/\lfloor \lambda/2 \rfloor \rceil$ λ-transposições são necessárias para posicionar todos os segmentos. □

Como um exemplo do lema anterior, seja $\pi = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$ uma permutação sem sinais e suponha que queremos obter $\pi \cdot \tau(1,5,10) = (5\ 6\ 7\ 8\ 9\ 1\ 2\ 3\ 4)$ com 5-transposições. Sejam $A = (1\ 2\ 3)$ e $B = (4)$ os subsegmentos do primeiro segmento da transposição (P) e sejam $C = (5\ 6)$, $D = (7\ 8)$, e $E = (9)$ subsegmentos do segundo segmento da transposição (S), como no lema. O processo descrito no lema primeiro troca B com C, gerando $(1\ 2\ 3\ 5\ 6\ 4\ 7\ 8\ 9)$, então B com D, gerando $(1\ 2\ 3\ 5\ 6\ 7\ 8\ 4\ 9)$, e então B com E, gerando $(1\ 2\ 3\ 5\ 6\ 7\ 8\ 9\ 4)$. Note que os segmentos C, D, e E estão consecutivamente começando depois de A. Os passos finais do processo são para trocar A com C, depois A com D, e A com E, gerando $(5\ 6\ 7\ 8\ 9\ 1\ 2\ 3\ 4) = \pi \cdot \tau(1,5,10)$. Assim, obtemos a permutação desejada com um total de $\lceil (j-i)/\lceil \lambda/2 \rceil \rceil \lceil (k-j)/\lfloor \lambda/2 \rfloor \rceil$ $= \lceil (5-1)/\lceil 5/2 \rceil \rceil \lceil (10-5)/\lfloor 5/2 \rfloor \rceil = \lceil 4/3 \rceil \lceil 5/2 \rceil = 6$ 5-transposições.

Os teoremas 1 até 5 usam os lemas 8 e 9 para obter algoritmos de aproximação para os 5 problemas que estamos considerando nesta seção.

Teorema 1. *O problema de Ordenação de Permutações sem Sinais por λ-Reversões tem um algoritmo de aproximação com fator* $0.6875p(p+1)$*, onde* $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.

Demonstração. Seja ALG_r o algoritmo de 1.375-aproximação para o problema de Ordenação de Permutações sem Sinais por Reversões [7]. Podemos criar um algoritmo para o problema de Ordenação de Permutações sem sinais por λ-Reversões que troca cada reversão dada por ALG_r para uma permutação π por uma sequência de λ-reversões, como descritas no Lema 8.

Já que uma reversão pode ter tamanho de no máximo n, ela será trocada por no máximo $\frac{p(p+1)}{2}$ λ-reversões.

Seja $\text{ALG}_r(\pi)$ o tamanho da sequência de ordenação produzida pelo ALG_r para ordenar uma permutação π. A quantidade de λ-reversões utilizadas pelo nosso algoritmo para ordenar π é assim no máximo $\text{ALG}_r(\pi)\frac{p(p+1)}{2} \leq 1.375 d_r(\pi)\frac{p(p+1)}{2} \leq 0.6875p(p+1)d_r^{\lambda}(\pi)$, onde a última inequação segue do Lema 7. □

Corolário 4. *O problema de Ordenação de Permutações sem Sinais por λ-Reversões tem um algoritmo de aproximação com fator 8.25 para todo $n > 3$ e $\lambda > \lceil n/2 \rceil$.*

Teorema 2. *O problema de Ordenação de Permutações com Sinais por λ-Reversões tem um algoritmo de aproximação com fator* $\frac{p(p+1)}{2}$*, onde* $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.

Demonstração. Similar à prova do Teorema 1, mas utilizando o algoritmo ótimo para o problema de Ordenação de Permutações por Reversões [28]. □

Corolário 5. *O problema de Ordenação de Permutações com Sinais por λ-Reversões tem um algoritmo de aproximação com fator 6 para todo $n > 3$ e $\lambda > \lceil n/2 \rceil$.*

Teorema 3. *O problema de Ordenação de Permutações por λ-Transposições tem um algoritmo de aproximação com fator* $1.375\left\lceil \frac{\lceil n/2 \rceil}{\lceil \lambda/2 \rceil} \right\rceil \left\lceil \frac{\lfloor n/2 \rfloor}{\lfloor \lambda/2 \rfloor} \right\rceil$.

Demonstração. Seja ALG_t o algoritmo de 1.375-aproximação para o problema de Ordenação de Permutações por Transposições [20]. Podemos criar um algoritmo para o problema de Ordenação de Permutações por λ-Transposições que troca cada transposição dada por ALG_t por uma sequência de λ-transposições, como descrito no Lema 9.

Já que uma transposição pode ter tamanho de no máximo n, ela será trocada por no máximo $T = \lceil \lceil n/2 \rceil / \lceil \lambda/2 \rceil \rceil \lceil \lfloor n/2 \rfloor / \lfloor \lambda/2 \rfloor \rceil$ λ-transposições.

Seja $\text{ALG}_t(\pi)$ o tamanho da sequência de ordenação produzida peloALG_t para ordenar uma permutação π. A quantidade de λ-transposições utilizadas pelo nosso algoritmo para ordenar π é assim no máximo $\text{ALG}_t(\pi)\, T \leq 1.375\, T\, d_t(\pi) \leq 1.375\, T\, d_t^{\lambda}(\pi)$, onde a última inequação segue do Lema 7. □

Corolário 6. *O problema de Ordenação de Permutações por λ-Transposições tem um algoritmo de aproximação com fator 5.5 para todo $n > 3$ e $\lambda > \lceil n/2 \rceil$.*

Teorema 4. *O problema de Ordenação de Permutações sem Sinais por λ-Reversões e λ-Transposições tem um algoritmo com fator de aproximação* $\alpha\, p(p+1)$*, onde* α *é o fator de aproximação do algoritmo de decomposição de ciclos para grafos de breakpoint e* $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.

Demonstração. Seja ALG_{rt} o algoritmo de 2α-aproximação para o problema de Ordenação de Permutações sem Sinais por Reversões e Transposições [43]. Nosso algoritmo para Ordenação de Permutações sem Sinais por λ-reversões e λ-transposições transforma a sequência de operações dada pelo ALG_{rt} em λ-operações, como descrito nos lemas 8 e 9.

Já que reversões e transposições podem ter tamanhos de no máximo n, cada transposição é trocada por no máximo $T = \lceil \lceil n/2 \rceil / \lceil \lambda/2 \rceil \rceil \lceil \lfloor n/2 \rfloor / \lfloor \lambda/2 \rfloor \rceil$ λ-transposições, e cada reversão é trocada por no máximo $R = p(p+1)/2$ λ-reversões. Observe que $T \leq R$ e assim podemos supor que, no pior caso, ALG_{rt} somente usa reversões para ordenar a permutação.

Seja $\mathrm{ALG}_{rt}(\pi)$ o tamanho da sequência de ordenação utilizada pelo ALG_{rt} para ordenar uma permutação π. A quantidade de operações utilizadas pelo nosso algoritmo é de no máximo $\mathrm{ALG}_{rt}(\pi) R \leq 2\,\alpha\, R\, d_{rt}(\pi) \leq \alpha\, p(p+1) d^{\lambda}_{rt}(\pi)$, onde a última inequação segue do Lema 7. □

Corolário 7. *O problema de Ordenação de Permutações sem Sinais por λ-Reversões e λ-Transposições tem um algoritmo de aproximação com fator* 12α *para todo* $n > 3$ *e* $\lambda > \lceil n/2 \rceil$.

Teorema 5. *O problema de Ordenação de Permutações com Sinais por λ-Reversões e λ-Transposições tem um algoritmo de aproximação com fator* $p(p+1)$*, onde* $p = \left\lceil \frac{n}{\lfloor \lambda/2 \rfloor} \right\rceil$.

Demonstração. Similar a prova do Teorema 4, mas utilizando o algoritmo de 2-aproximação para o problema de Ordenação de Permutações com Sinais por Reversões e Transposições [49]. □

Corolário 8. *O problema de Ordenação de Permutações por λ-Reversões e λ-Transposições tem um algoritmo de aproximação com fator* 12 *para todo* $n > 3$ *e* $\lambda > \lceil n/2 \rceil$.

4.2 Algoritmos de Aproximação para Valores Pequenos de λ

Nesta seção apresentamos algoritmos que possuem melhores fatores de aproximação para valores pequenos de λ.

4.2.1 Algoritmos Baseados em Entropia

Seja π uma permutação com ou sem sinais. Seja $\Delta_{\mathrm{ent}}(\pi, \sigma) = \mathrm{ent}(\pi \cdot \sigma) - \mathrm{ent}(\pi)$ a variação de entropia depois da aplicação de uma operação σ. Note que para calcular $\Delta_{\mathrm{ent}}(\pi, \sigma)$ é suficiente determinar a entropia dos elementos afetados por σ (a entropia dos outros elementos não muda).

Observe que $\text{ent}(\iota) = 0$ e $\text{ent}(\pi) > 0$ para todas permutações sem sinais $\pi \neq \iota$. O Lema 10 dá um limitante superior sobre a variação de entropia causada por qualquer λ-reversão ou λ-transposição. Ele implica no Corolário 9, que apresenta limitantes inferiores para as distâncias de ordenação dos problemas de ordenação de permutações sem sinais abordados neste capítulo.

Lema 10. *Seja $\Delta_{\text{ent}}^{max}(\pi, \sigma)$ a máxima variação de entropia causada por uma λ-reversão ou por uma λ-transposição σ aplicada sobre uma permutação π com ou sem sinais. Então temos $\Delta_{\text{ent}}^{max}(\pi, \sigma) = 2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor$.*

Demonstração. Primeiramente, seja σ uma λ-reversão $\rho(i, j)$. Note que a máxima variação de entropia ocorre quando $j - i + 1 = \lambda$, que é o momento em que o maior número de elementos estão envolvidos. Neste caso, depois da reversão, o elemento π_i termina $\lambda - 1$ posições distante de i, o elemento π_{i+1} termina $\lambda - 3$ posições distante de $i+1$, e assim por diante, até o elemento $\pi_{i+\lfloor \lambda/2 \rfloor - 1}$, que termina 1 posição distante de $i + \lfloor \lambda/2 \rfloor - 1$ se λ é par ou termina 2 posições distante se λ é impar. Similarmente, o elemento π_j termina $\lambda - 1$ posições distante de j, o elemento π_{j-1} termina $\lambda - 3$ posições distante de $j - 1$, e assim por diante, até o elemento $\pi_{j-\lfloor \lambda/2 \rfloor + 1}$, que similarmente termina 1 ou 2 posições distante de $j - \lfloor \lambda/2 \rfloor + 1$, se λ é par ou impar, respectivamente. Quando λ é impar, o elemento $i + \lceil \lambda/2 \rceil$ permanece na sua posição. Desta forma, a máxima variação de entropia para cada elemento entre as posições i e j é no máximo $2\sum_{\ell=1}^{\lfloor \lambda/2 \rfloor}(\lambda - (2\ell - 1)) = 2\lfloor \lambda/2 \rfloor \lceil \lambda/2 \rceil$.

Agora seja σ uma λ-transposição $\tau(i, j, k)$ e, similarmente, suponha $k - i = \lambda$, quando podemos ter a variação máxima de entropia. Neste caso, depois da transposição, todos elementos π_h, para $i \leq h < j$, terminam $k - j$ posições distantes de h, enquanto todos elementos π_ℓ, para $j \leq \ell < k$, terminam $j - i$ posições distantes de ℓ.

Uma vez que há $j - i$ elementos do primeiro tipo e $k - j$ elementos do segundo tipo, a máxima variação de entropia para cada elemento entre i e k é no máximo $2(j-i)(k-j) \leq 2\lfloor \lambda/2 \rfloor \lceil \lambda/2 \rceil$. □

Corolário 9. *Para qualquer permutação π com ou sem sinais, $\lambda \geq 2$, e $\beta \in \{r, t, rt, \bar{r}, \bar{r}t\}$, temos $d_\beta^\lambda(\pi) \geq \text{ent}(\pi)/(2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor)$.*

Denote por $\text{neg}(\pi)$ o número de elementos negativos de uma permutação com sinais π e observe que a única permutação π com $\text{neg}(\pi) + \text{ent}(\pi) = 0$ é a permutação identidade. Seja $\text{score}_{ent}(\pi, \sigma) = ((\text{ent}(\pi) - \text{ent}(\pi \cdot \sigma)) + (\text{neg}(\pi) - \text{neg}(\pi \cdot \sigma))$ a pontuação de entropia de uma λ-operação σ aplicada sobre π. Note que, se π é uma permutação sem sinais, então $\text{score}_{ent}(\pi, \sigma) = \text{ent}(\pi) - \text{ent}(\pi \cdot \sigma)$ é a quantidade de entropia decrementada após aplicar σ. O Lema 11 dá um limitante superior para a pontuação de entropia de qualquer λ-reversão ou λ-transposição. Ele implica no Corolário 10, que apresenta limitantes inferiores para a distância de ordenação para os problemas de ordenação de permutações com sinais que estamos abordando neste capítulo.

Lema 11. *Seja $\text{score}_{ent}^{max}(\pi, \sigma)$ a maior pontuação de entropia possível de uma λ-operação σ aplicada sobre uma permutação com sinais π. Temos $\text{score}_{ent}^{max}(\pi, \sigma) = 2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor + \lambda$.*

Demonstração. Pelo Lema 10, temos que a máxima variação de entropia de uma λ-operação é $2\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor$. Por definição, uma λ-operação σ envolve no máximo λ elementos

de uma permutação e portanto no máximo λ deles podem ter seus sinais trocados para positivo. Logo, $\text{score}^{max}_{ent}(\pi, \sigma) = 2\lceil\lambda/2\rceil\lfloor\lambda/2\rfloor + \lambda$. □

Corolário 10. *Para qualquer permutação com sinais π, $\lambda \geq 2$, e $\beta \in \{\bar{r}, \bar{r}t\}$, temos $d^{\lambda}_{\beta}(\pi) \geq (\text{ent}(\pi) + \text{neg}(\pi))/((2\lceil\lambda/2\rceil\lfloor\lambda/2\rfloor) + \lambda)$.*

Seja π uma permutação (com ou sem sinais) de tamanho n e sejam i e j dois inteiros tais que $1 \leq i < j \leq n$. A operação $\phi(\pi, i, j)$ devolve uma permutação π' tal que $\pi'_i = \pi_j$, $\pi'_j = \pi_i$, e $\pi'_k = \pi_k$ para todo $k \notin \{i, j\}$. Em outras palavras, somente os elementos π_i e π_j são trocados, e se π é uma permutação com sinais, o sinal de todos elementos de π permanecem os mesmos em π'. Os lemas 12, 13, e 14 mostram como obter $\phi(\pi, i, j)$ com apenas λ-reversões e λ-transposições, respectivamente.

Lema 12. *Seja π uma permutação sem sinais, $\lambda \geq 2$, e sejam i e j posições tais que $1 \leq i < j \leq n$. É possível obter $\phi(\pi, i, j)$ aplicando no máximo $2x$ λ-reversões sobre π, onde $x = \lceil \frac{j-i}{\lambda-1} \rceil$.*

Demonstração. Mostramos que o resultado segue ao considerar dois casos, de acordo com a relação entre i e j. Se $j-i \leq \lambda-1$, então no máximo duas λ-reversões são necessárias como descrevemos. Primeiro aplicamos a operação $\rho(i, j)$ que troca a posição dos elementos π_i e π_j. É fácil ver que, se $j-i \leq 2$, nós já obtemos $\phi(\pi, i, j)$ com somente esta operação. Caso contrário, observe que teremos o segmento $\pi_{i+1}, \ldots, \pi_{j-1}$ na ordem reversa (de acordo com $\phi(\pi, i, j)$). Assim, temos que aplicar uma segunda operação $\rho(i+1, j-1)$ para revertê-lo novamente e, com isso, obtemos $\phi(\pi, i, j)$ utilizando duas λ-reversões.

Caso contrário, $j - i > \lambda - 1$. Num primeiro passo, o elemento π_i é movido para a posição j incrementando repetidamente sua posição em $\lambda - 1$ (exceto, talvez, pelo último movimento) com exatamente $x = \lceil (j - i)/(\lambda - 1) \rceil$ λ-reversões aplicadas sucessivamente. Formalmente, isto é feito ao aplicar a sequência de λ-reversões $\rho(i, i + (\lambda - 1))$, $\rho(i + (\lambda - 1), i + 2(\lambda - 1))$, $\rho(i + 2(\lambda - 1), i + 3(\lambda - 1))$, $\ldots$, $\rho(i + (x - 1)(\lambda - 1), j)$. Agora o elemento π_j está na posição $i + (x-1)(\lambda-1)$ e os elementos π_t, para $i < t < j$, não estão necessariamente na posição t. Para corrigir isto e, ao mesmo tempo, mover o elemento π_j para a posição i, temos um segundo passo que aplica uma sequência com as mesmas λ-reversões utilizadas anteriormente (exceto pela última delas), mas na ordem reversa, para repetidamente decrementar a posição do elemento π_j em $\lambda - 1$. Assim, um total de $x - 1$ operações extras são necessárias. Formalmente, a sequência é $\rho(i + (x-2)(\lambda-1), i + (x-1)(\lambda-1))$, $\rho(i + (x-3)(\lambda-1), i + (x-2)(\lambda-1)), \ldots, \rho(i + 2(\lambda-1), i + 3(\lambda-1))$, $\rho(i + (\lambda-1), i + 2(\lambda-1))$, $\rho(i, i + (\lambda-1))$. Neste ponto, se o tamanho da λ-reversão $\rho(i + (x-1)(\lambda-1), j)$ (a última λ-reversão do primeiro passo) for menor que ou igual a 3, então ao final do processo teremos diretamente obtido $\phi(\pi, i, j)$. Caso contrário, temos que aplicar uma λ-reversão $\rho(i + (x-1)(\lambda-1) + 1, j-1)$ a mais para obter $\phi(\pi, i, j)$, o que totalizam $2x$ λ-reversões. □

Como um exemplo para o Lema 12, seja π = (7 2 3 4 5 6 1) uma permutação sem sinais e suponha que nosso objetivo é obter $\phi(\pi, 1, 7)$ = (1 2 3 4 5 6 7). Então, $i = 1$ e $j = 7$. Seja $\lambda = 4$. Note que $x = \lceil (7 - 1)/(4 - 1) \rceil = 2$. Seguindo a prova, temos $j - i = 6 > 3 = \lambda - 1$ e assim as seguintes reversões (sublinhadas em cada permutação)

são utilizadas:

$$\begin{aligned}
\pi &= (\underline{7\ 2\ 3\ 4}\ 5\ 6\ 1) \cdot \rho(1,4) \\
\pi^1 &= (4\ 3\ 2\ \underline{7\ 5\ 6\ 1}) \cdot \rho(4,7) \\
\pi^2 &= (\underline{4\ 3\ 2\ 1}\ 6\ 5\ 7) \cdot \rho(1,4) \\
\pi^3 &= (1\ 2\ 3\ 4\ \underline{6\ 5}\ 7) \cdot \rho(5,6) \\
\pi^4 &= (1\ 2\ 3\ 4\ 5\ 6\ 7)
\end{aligned}$$

No exemplo, note que as 4-reversões $\rho(1,4)$ e $\rho(4,7)$ foram utilizadas para posicionar o elemento $\pi_1 = 7$ na posição 7, e que somente a 4-reversão $\rho(1,4)$ é suficiente para posicionar $\pi_7 = 1$ na posição 1. Como descrito na prova, se a última operação da primeira etapa (neste exemplo, a operação $\rho(4,7)$) possui tamanho maior do que 3, a reversão $\rho(5,6)$ é necessária para que os elementos 5 e 6 não fiquem em ordem reversa.

Lema 13. *Seja π uma permutação com sinais, $\lambda \geq 2$, e sejam i e j posições tais que $1 \leq i < j \leq n$. É possível obter $\phi(\pi, i, j)$ aplicando no máximo $2x + 2$ λ-reversões sobre π, onde $x = \left\lceil \frac{j-i}{\lambda-1} \right\rceil$.*

Demonstração. Mostramos que o resultado segue ao considerar dois casos, de acordo com a relação entre i e j. Se $j - i \leq \lambda - 1$, então no máximo quatro λ-reversões são necessárias como descrevemos. Primeiro aplicamos as três operações $\rho(i,j)$, $\rho(i,i)$ e $\rho(j,j)$ que trocam as posições dos elementos π_i e π_j e então corrige seus sinais (de acordo com $\phi(\pi,i,j)$). É fácil ver que, se $j - i \leq 1$, nós já obtemos $\phi(\pi,i,j)$ com três operações. Caso contrário, observe que teremos os elementos do segmento $\pi_{i+1}, \ldots, \pi_{j-1}$ na ordem reversa e seus sinais opostos (de acordo com $\phi(\pi,i,j)$). Assim, temos que aplicar uma operação extra $\rho(i+1, j-1)$ para corrigir isto e, daí, obtemos $\phi(\pi,i,j)$ com quatro operações.

Caso contrário, $j - i > \lambda - 1$. Seja $\pi' = \phi(\pi,i,j)$. Inicialmente, aplicamos as (no máximo) $2x$ λ-reversões descritas na prova do Lema 12 para obter uma permutação π'' tal que $|\pi''_t| = |\pi'_t|$ para todo $i \leq t \leq j$. Já que o Lema 12 aplica as mesmas λ-reversões duas vezes para os elementos $\pi_{t'}$ onde $i < t' < i + (x-1)(\lambda-1)$, nós na verdade teremos $\pi''_{t'} = \pi'_{t'}$ para todos tais t'. Também, é fácil ver que se x for um número par, então teremos $\pi''_i = \pi'_i$ e $\pi''_j = \pi'_j$. Caso contrário, duas extras λ-reversões unitárias $\rho(i,i)$ e $\rho(j,j)$ tem que ser aplicadas para corrigir os sinais de π_i e π_j.

Olhamos então os elementos $\pi''_{(i+(x-1)(\lambda-1))}, \ldots, \pi''_{j-1}$. Observe que, se exatamente $2x$ λ-reversões foram usadas pelo Lema 12, então estes elementos também são revertidos duas vezes e teremos $\pi'' = \pi'$ com no máximo $2x + 2$ operações. Caso contrário, significa que $2x - 1$ λ-reversões foram utilizadas para obter π'' e a λ-reversão $\rho(i + (x-1)(\lambda-1), j)$ (a última operação da sequência de λ-reversões descritas no primeiro passo da prova do Lema 12) teve tamanho menor ou igual a 3. Se o tamanho foi igual a 3, essa λ-reversão afetou os elementos π_i, π_{j-1} e π_j, então apenas temos que aplicar uma reversão unitária extra $\rho(j-1, j-1)$ para obter $\pi'' = \pi'$, o que resulta em no máximo $2x + 2$ λ-reversões. Caso contrário, o tamanho de $\rho(i + (x-1)(\lambda-1), j)$ foi igual a 2, então somente os elementos π_i e π_j foram envolvidos pela operação. Assim, temos $\pi'' = \pi'$ com um total de no máximo $2x + 1$ λ-reversões. □

Lema 14. *Seja π uma permutação (com ou sem sinais), $\lambda \geq 2$, e i e j posições tais que $1 \leq i < j \leq n$. É possível obter $\phi(\pi, i, j)$ aplicando $x + y$ λ-transposições sobre π, onde $x = \left\lceil \frac{j-i}{\lambda-1} \right\rceil$ e $y = \left\lceil \frac{j-i-1}{\lambda-1} \right\rceil$.*

Demonstração. Mostramos que o resultado segue considerando dois casos, de acordo com a relação entre i e j. Se $j - i \leq \lambda - 1$, então no máximo duas λ-transposições são necessárias como descrevemos. Primeiro aplicamos $\tau(i, i+1, j+1)$ que coloca o elemento π_i na posição j. Quando $j = i+1$, isto também coloca o elemento π_j na posição i e, já que não há elementos entre π_i e π_j, temos que a permutação $\phi(\pi, i, j)$ já foi alcançada. Caso contrário, observe que teremos os elementos π_t, para $i < t \leq j$, exatamente uma posição à esquerda de suas posições originais em π. Assim, temos que aplicar uma segunda operação $\tau(i, j-1, j)$ a fim de corrigir isto. Note que, depois de aplicar esta segunda operação, temos π_j na posição i ao mesmo tempo em que os elementos $i < t < j$ foram movidos uma posição para a direita e, então, nós alcançamos $\phi(\pi, i, j)$ com duas λ-transposições.

Caso contrário, $j - i > \lambda - 1$. Inicialmente nós movemos o elemento π_i para a posição j ao repetidamente incrementar sua posição em $\lambda - 1$ (exceto, talvez, no último movimento) com exatamente $x = \lceil (j-i)/(\lambda-1) \rceil$ λ-transposições aplicadas sucessivamente. Formalmente, isto é feito ao aplicar a sequência de λ-transposições $\tau(i, i+1, i+\lambda)$, $\tau(i+(\lambda-1), i+(\lambda-1)+1, i+2(\lambda-1))$, $\tau(i+2(\lambda-1), i+2(\lambda-1)+1, i+2(\lambda-1)+\lambda)$, $\ldots$, $\tau(i+(x-1)(\lambda-1), i+(x-1)(\lambda-1)+1, j+1)$. Posteriormente, cada elemento π_t, para $i < t \leq j$, está exatamente uma posição à esquerda de sua posição original em π. Para corrigir isto, podemos aplicar uma sequência similar de λ-transposições, mas agora nós repetidamente decrementamos a posição de π_j (que agora está na posição $j-1$) em $\lambda - 1$ (exceto, talvez, na última operação) com exatamente $y = \lceil (j-1-i)/(\lambda-1) \rceil$ λ-transposições aplicadas sucessivamente. Formalmente, a sequência é $\tau(j-\lambda, j-1, j)$, $\tau(j-(\lambda-1)-\lambda, j-(\lambda-1)-1, j-(\lambda-1))$, $\tau(j-2(\lambda-1)-\lambda, j-2(\lambda-1)-1, j-2(\lambda-1))$, $\ldots$, $\tau(i, j-y(\lambda-1)-1, j-y(\lambda-1))$. Note que cada λ-transposição move $\lambda - 1$ (novamente, exceto, talvez, pela última operação) elementos π_t, para $i < t < j$, uma posição à direita ao trocar todos eles com o elemento π_j. No final desse processo, nós diretamente temos $\phi(\pi, i, j)$. □

O Lema 15 é auxiliar ao Lema 16, que mostra que é sempre possível reduzir a entropia de qualquer permutação com a operação $\phi(\pi, i, j)$.

Lema 15. *Para toda permutação (com ou sem sinais) com* $\mathrm{ent}(\pi) > 0$*, existe um par de elementos π_i e π_j, com $1 \leq i < j \leq n$, tal que $\pi_i \geq j$ e $\pi_j \leq i$.*

Demonstração. Seja G_π o grafo direcionado tal que $V(G_\pi) = \{1, 2, \ldots, n\}$ e $E(G_\pi) = \{(|\pi_i|, i) \colon 1 \leq i \leq n\}$. Note que cada vértice tem grau de entrada 1 e grau de saída 1, e, portanto, os componentes de G_π são ciclos. Note também que somente G_ι tem n ciclos unitários.

Seja C um ciclo qualquer de G_π com pelo menos dois vértices e seja u o vértice de menor valor de C. Seja $B = (v_1, v_2, \ldots, v_\ell)$ uma sequência maximal de vértices de C tal que $v_1 = u$, $v_i < v_{i+1}$ para todo $1 \leq i < \ell$, e $(v_i, v_{i+1}) \in E(G_\pi)$.

Já que os vértices de B estão em um ciclo e B é maximal, a aresta incidente a v_ℓ é da forma (v_ℓ, x), com $x < v_\ell$. Se $v_{\ell-1} \leq x$, então considere $i = x$ e $j = v_\ell$. Neste caso, temos

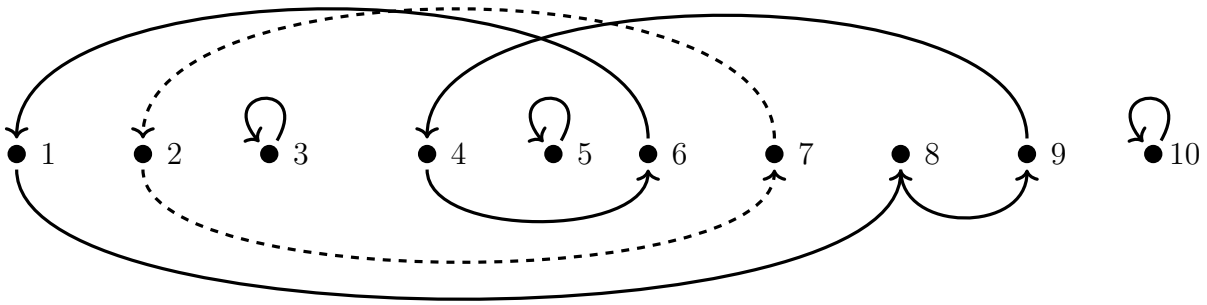


Figura 4.1: Grafo G_π para $\pi = (6\ 7\ 3\ 9\ 5\ 4\ 2\ 1\ 8\ 10)$. Considerando a notação da prova do Lema 15, se $C = (1, 8, 9, 4, 6, 1)$, então temos $B = (1, 8, 9)$.

$|\pi_i| = |\pi_x| = v_\ell = j$ e $|\pi_j| = |\pi_{v_\ell}| = v_{\ell-1} \leq x = i$ e então o lema segue. Se $v_{\ell-1} > x$, então seja k, com $1 \leq k < \ell - 1$, tal que $v_k \leq x < v_{k+1}$ e considere $i = x$ e $j = v_{k+1}$. Neste caso, temos $|\pi_i| = |\pi_x| = v_\ell > v_{k+1} = j$ e $|\pi_j| = |\pi_{v_{k+1}}| = v_k \leq x = i$ e o lema segue. Veja o exemplo da Figura 4.1. □

Lema 16. *Para toda permutação (com ou sem sinais)* π *com* $\mathrm{ent}(\pi) > 0$*, é possível obter outra permutação* $\phi(\pi, i, j)$ *tal que* $\mathrm{ent}(\phi(\pi, i, j)) = \mathrm{ent}(\pi) - 2(j - i)$*, para algum par* $1 \leq i < j \leq n$*.*

Demonstração. Sejam i e j como no Lema 15. Por definição, $\mathrm{ent}(\pi) - \mathrm{ent}(\phi(\pi, i, j)) = ||\pi_i| - i| + ||\pi_j| - j| - ||\pi_i| - j| - ||\pi_j| - i|$. Já que $|\pi_i| \geq j > i$, temos $||\pi_i| - j| = |\pi_i| - j$ e $||\pi_i| - i| = |\pi_i| - i$. Já que $|\pi_j| \leq i < j$, temos $||\pi_j| - i| = i - |\pi_j|$ e $||\pi_j| - j| = j - |\pi_j|$. Assim, $\mathrm{ent}(\pi) - \mathrm{ent}(\phi(\pi, i, j)) = |\pi_i| - i + j - |\pi_j| - |\pi_i| + j - i + |\pi_j| = 2(j - i)$. □

Uma permutação $\pi \neq \iota$ é chamada *normal* se existe pelo menos uma λ-operação σ tal que $\mathrm{ent}(\pi) > \mathrm{ent}(\pi \cdot \sigma)$ e $\mathrm{score}_{ent}(\pi, \sigma) > 0$. Além disso, já que a entropia de qualquer permutação é um número par, temos $\mathrm{score}_{ent}(\pi, \sigma) \geq 2$ e $\mathrm{score}_{ent}(\pi, \sigma) \geq 1$ para permutações normais π sem sinais e com sinais, respectivamente. Chamamos π de *especial* caso não seja normal.

Considere a permutação normal com sinais $\pi = (-4\ +5\ -2\ -3\ +1)$ em que $\mathrm{ent}(\pi) = 12$ e $\mathrm{neg}(\pi) = 3$. Note que $\mathrm{score}_{ent}(\pi, \rho(2, 3)) = 2$, mas $\mathrm{score}_{ent}(\pi, \rho(1, 5)) = 9$. O seguinte algoritmo é guloso em termos de sempre escolher uma λ-operação que possui a maior pontuação de entropia quando aplicada a uma permutação normal com ou sem sinais. Quando a permutação é especial, o Lema 16 é aplicado. Note que, em ambos os casos, o algoritmo sempre reduz a quantidade de entropia até eventualmente alcançar uma permutação γ tal que $\mathrm{ent}(\gamma) = 0$. Para os problemas de ordenação de permutação sem sinais, neste ponto nós já temos $\gamma = \iota$ e, para os problemas de ordenação de permutações com sinais, o algoritmo aplica mais $\mathrm{neg}(\gamma)$ reversões unitárias para alcançar ι. O algoritmo recebe um modelo de rearranjo β, para qualquer um dos problemas que estamos considerando, e ele executa em tempo polinomial, como discutimos posteriormente. Nós mostramos que o algoritmo garante fator de aproximação no Teorema 6 e no Teorema 7, para os problemas envolvendo permutações sem sinais e com sinais, respectivamente.

```
1: function AlgoritmoGuloso(π, λ, β)
2:     while ent(π) > 0 do
3:         if π é uma permutação normal then
4:             Seja σ uma λ-operação com score_ent(π, σ) máximo e ent(π) > ent(π · σ)
5:             π ← π · σ
6:         else
7:             Sejam i e j posições tais que |π_i| ≥ j, |π_j| ≤ i, e 1 ≤ i < j ≤ n
8:             π ← φ(π, i, j), de acordo com o Lema 12, 13 ou 14
9:     if π é uma permutação com sinais then
10:        Aplique uma reversão unitária ρ(i, i) em cada elemento π_i < 0
```

Teorema 6. *Os problemas de Ordenação de Permutações sem Sinais por λ-Reversões, por λ-Transposições, ou por ambas as operações têm algoritmo de aproximação de fator* $4\lceil\lambda/2\rceil\lfloor\lambda/2\rfloor$.

Demonstração. Se π é uma permutação normal, então o algoritmo escolhe uma λ-operação σ tal que $\text{score}_{ent}(\pi,\sigma) = \text{ent}(\pi) - \text{ent}(\pi\cdot\sigma) \geq 2$.

Caso contrário, π é especial. Sejam i e j como no AlgoritmoGuloso e note que $\pi' = \phi(\pi,i,j)$ é uma permutação com $\text{ent}(\pi') = \text{ent}(\pi) - 2(j-i)$ (segundo o Lema 16), obtida através das operações descritas no Lema 12 ou no Lema 14.

Se $j - i \leq \lambda - 1$, então obtemos π' ao aplicar no máximo duas λ-operações. Logo, a pontuação de entropia por operação é pelo menos $(2(j-i))/2 \geq 1$, para qualquer um dos 3 problemas de ordenação de permutações sem sinais que estamos abordando.

Se $j-i > \lambda-1$, então obtemos π' ao aplicar no máximo $2\lceil(j-i)/(\lambda-1)\rceil$ λ-reversões ou $\lceil(j-i)/(\lambda-1)\rceil + \lceil(j-i-1)/(\lambda-1)\rceil$ λ-transposições. Já que $2\lceil(j-i)/(\lambda-1)\rceil \geq \lceil(j-i)/(\lambda-1)\rceil + \lceil(j-i-1)/(\lambda-1)\rceil$, temos que a pontuação de entropia obtida por operação é pelo menos $(2(j-i))/(2\lceil\frac{j-i}{\lambda-1}\rceil) = (j-i)/(\lceil\frac{j-i}{\lambda-1}\rceil) \geq (j-i)/(2\frac{j-i}{\lambda-1}) = \frac{\lambda-1}{2} \geq \frac{1}{2}$.

Assim, no pior caso, temos $1/2$ de pontuação de entropia por operação para todos os 3 problemas de ordenação de permutações sem sinais, o que significa que a quantidade total de operações utilizadas pelo AlgoritmoGuloso é no máximo $\frac{\text{ent}(\pi)}{1/2} = 2\,\text{ent}(\pi) \leq 2\,d_\beta^\lambda(\pi)\,2\lceil\lambda/2\rceil\lfloor\lambda/2\rfloor$, onde a inequação segue do Lema 10 e $\beta \in \{r,t,rt\}$. □

Teorema 7. *Para qualquer permutação com sinais* π *com* $\text{ent}(\pi) \neq 0$*, os problemas de Ordenação de Permutações com Sinais por λ-Reversões e por λ-Reversões e λ-Transposições têm algoritmo de aproximação de fator* $8\lceil\lambda/2\rceil\lfloor\lambda/2\rfloor + \lambda$.

Demonstração. Se π é uma permutação normal, então o algoritmo escolhe uma λ-operação σ com $\text{score}_{ent}(\pi,\sigma) \geq 1$.

Caso contrário, sejam i e j como no AlgoritmoGuloso e note que $\pi' = \phi(\pi,i,j)$ é uma permutação com $\text{ent}(\pi') = \text{ent}(\pi) - 2(j-i)$ (segundo o Lema 16), obtida através das operações descritas nos lemas 13 e 14 e então $\text{neg}(\pi') = \text{neg}(\pi)$ neste caso.

Se $j - i \leq \lambda - 1$, então obtemos π' ao aplicar no máximo 4 λ-operações. Logo, a pontuação de entropia obtida por operação é pelo menos $(2(j-i))/4 \geq 2/4 = 1/2$.

Se $j-i > \lambda-1$, então obtemos π' ao aplicar no máximo $2\lceil(j-i)/(\lambda-1)\rceil+2$ λ-reversões ou $\lceil(j-i)/(\lambda-1)\rceil + \lceil(j-i-1)/(\lambda-1)\rceil$ λ-transposições. Já que $2\lceil(j-i)/(\lambda-1)\rceil + 2 \geq$

$\lceil (j-i)/(\lambda-1) \rceil + \lceil (j-i-1)/(\lambda-1) \rceil$, temos que a pontuação de entropia por operação é pelo menos $(2(j-i))/(2\lceil \frac{j-i}{\lambda-1} \rceil + 2) \geq (2(j-i))/(4\lceil \frac{j-i}{\lambda-1} \rceil) \geq (j-i)/(4\frac{j-i}{\lambda-1}) = \frac{\lambda-1}{4} \geq \frac{1}{4}$.

Assim, no pior caso, temos $1/4$ de pontuação de entropia por operação para ambos os problemas de ordenação de permutações com sinais. Quando obtemos uma permutação γ com $\text{ent}(\gamma) = 0$, o algoritmo aplica $\text{neg}(\gamma)$ λ-reversões unitárias a mais para obter a identidade. Assim a quantidade total de operações utilizadas pelo AlgoritmoGuloso é no máximo

$$\begin{aligned}
&\frac{(\text{ent}(\pi) + \text{neg}(\pi)) - (\text{ent}(\gamma) + \text{neg}(\gamma))}{1/4} + \text{neg}(\gamma) \\
&= 4\,(\text{ent}(\pi) + \text{neg}(\pi) - \text{neg}(\gamma)) + \text{neg}(\gamma) \\
&= 4\,(\text{ent}(\pi) + \text{neg}(\pi)) - 4\,\text{neg}(\gamma) + \text{neg}(\gamma) \\
&\leq 4\,(\text{ent}(\pi) + \text{neg}(\pi)) \leq (8\,\lceil \lambda/2 \rceil \lfloor \lambda/2 \rfloor + \lambda)\, d_{\beta}^{\lambda}(\pi),
\end{aligned}$$

onde a última inequação segue do Lema 10 e $\beta \in \{\bar{r}, \bar{r}t\}$. □

O Lema 17 e o Teorema 8 mostram que o algoritmo garante fator de aproximação constante para todas as permutações com sinais $\pi \neq \iota$ tais que $\text{ent}(\pi) = 0$. Eles utilizam a definição de *breakpoints*, apresentada na Seção 2.6.

Lema 17. *Para todas permutações com sinais* $\pi \neq \iota$ *com* $\text{ent}(\pi) = 0$ *e* $\lambda \geq 2$*, temos* $d_{\bar{r}}^{\lambda}(\pi) \geq \frac{\text{neg}(\pi)+1}{2}$ *e* $d_{\bar{r}t}^{\lambda}(\pi) \geq \frac{\text{neg}(\pi)+1}{3}$.

Demonstração. Observe que todas as permutações com sinais $\pi \neq \iota$ com $\text{ent}(\pi) = 0$ possuem $\text{neg}(\pi) = b(\pi) - 1$. A partir disso, o lema segue diretamente pelo limitante inferior dado pelo Lema 1. □

Teorema 8. *Para todas permutações com sinais* π *com* $\text{ent}(\pi) = 0$*, o* AlgoritmoGuloso *garante fator de aproximação* 2 *e* 3 *para os problemas de Ordenação de Permutações com Sinais por* λ*-Reversões e por* λ*-Reversões e* λ*-Transposições, respectivamente.*

Demonstração. Quando $\text{ent}(\pi) = 0$, o algoritmo apenas aplica $\text{neg}(\pi)$ reversões unitárias para alcançar ι. Pelo limitante inferior do Lema 17, temos $\text{neg}(\pi) \leq 2d_{\bar{r}}(\pi) - 1 \leq 2d_{\bar{r}}(\pi)$ e $\text{neg}(\pi) \leq 3d_{\bar{r}t}(\pi) - 1 \leq 3d_{\bar{r}t}(\pi)$. □

Em relação a complexidade dos algoritmos, primeiro observe que o primeiro laço "enquanto" na primeira linha executa $O(n^2)$ vezes uma vez que $\text{ent}(\pi) + \text{neg}(\pi)$ é $O(n^2)$ e, como visto nos teoremas 6 e 7, uma operação no pior caso tem pontuação de entropia pelo menos $1/2$ ou $1/4$. No comando "se" devemos decidir se uma permutação é normal, o que pode ser feito procurando a λ-operação desejada. Para isso, podemos simplesmente testar todas as possíveis λ-operações, o que leva tempo $O(n\lambda)$ para λ-reversões e $O(n\lambda^2)$ para λ-transposições. Além disso, leva tempo $O(\lambda)$ para calcular a pontuação de entropia de uma λ-permutação. Portanto, testar se uma permutação é normal leva tempo $O(n\lambda^3)$. No comando "se não", devemos primeiro obter as posições i e j como desejado, o que leva tempo $O(n)$. Agora considere obter $\phi(\pi, i, j)$ com λ-transposições. Pelo Lema 14, podemos usar no máximo $\lceil (j-i)/(\lambda-1) \rceil + \lceil (j-i-1)/(\lambda-1) \rceil$ tais operações, cada uma de tamanho no máximo λ. Isso implica que o tempo para executar cada transposição é $O(\lambda)$ e assim o tempo total para obter $\phi(\pi, i, j)$ é no máximo $O(\lambda)\left(\frac{j-i+1}{\lambda-1} + \frac{j-i}{\lambda-1}\right) \leq O(\lambda)2\frac{n}{\lambda-1} = O(n\lambda)$,

já que $\lambda \geq 2$. De forma similar, o tempo para obter $\phi(\pi, i, j)$ com reversões também é $O(n\lambda)$. Para permutações com sinais, no máximo $\text{neg}(\pi) \leq n$ reversões unitárias a mais são necessárias, levando tempo $O(n)$. Logo, o tempo total do AlgoritmoGuloso é $O(n^3\lambda^3)$ para qualquer um dos problemas que estamos considerando. Note que isto é polinomial, pois $\lambda = O(n)$.

4.2.2 Algoritmos Baseados em Inversões para Permutações sem Sinais

Nesta seção apresentamos algoritmos baseados na definição de inversões, apresentada na Seção 2.9. Inicialmente, mostramos no Lema 18 que sempre existe uma inversão em uma permutação $\pi \neq \iota$. Então, utilizando o limitante inferior apresentado no Corolário 2, mostramos um algoritmo genérico para ordenar permutações sem sinais que garante fator de aproximação quadrático para todos os problemas de ordenação de permutações sem sinais por λ-operações que estamos abordando neste capítulo. Este algoritmo é guloso, pois sempre escolhe a operação que reduz o máximo de inversões possíveis em cada passo do processo de ordenação.

Lema 18. *Para todas permutações π tais que* $\text{Inv}(\pi) > 0$ *existe uma inversão* (π_i, π_{i+1}).

Demonstração. Seja $\pi_1, \pi_2, \ldots, \pi_i$ uma subsequência maximal tal que $|\pi_1| < |\pi_2| < \ldots < |\pi_i|$. Já que $\text{Inv}(\pi) > 0$, temos $i < n$ e, assim, $|\pi_i| > |\pi_{i+1}|$. Logo (π_i, π_{i+1}) é uma inversão. □

Teorema 9. *Existe algoritmo de $\frac{\lambda(\lambda-1)}{2}$-aproximação para os problemas de Ordenação de Permutações sem Sinais por λ-Reversões, por λ-Transposições e por λ-Reversões e λ-Transposições.*

Demonstração. Seja $\lambda \geq 2$ um inteiro e seja $\pi \neq \iota$ uma permutação sem sinais. Já que a única permutação com $\text{Inv}(\pi) = 0$ é a identidade, podemos projetar um algoritmo guloso que repetidamente escolhe e aplica uma λ-operação σ tal que $\text{Inv}(\pi \cdot \sigma)$ é o menor possível e então ele irá, ao final, ordenar a permutação. Pelo Lema 18, sempre existe uma inversão (π_i, π_{i+1}) em π e, então, nosso algoritmo reduz pelo menos uma inversão por vez. Logo, o número de operações necessárias por tal algoritmo guloso é no máximo $\text{Inv}(\pi)$ e seu fator de aproximação segue diretamente do Corolário 2. □

Note que $\text{Inv}(\pi)$ é $O(n^2)$ e que $O(\lambda^2)$ λ-reversões e/ou $O(\lambda^3)$ λ-transposições tem que ser consideradas para encontrar a melhor operação no passo guloso do algoritmo. Já que podemos calcular a variação no número de inversões em tempo $O(\lambda\sqrt{\log(\lambda)})$ [13], a complexidade de tempo do algoritmo para o problema de ordenação de permutações sem sinais por λ-reversões e para os outros dois problemas abordados nesta seção é $O(n^2\lambda^3\sqrt{\log(\lambda)})$ e $O(n^2\lambda^4\sqrt{\log(\lambda)})$, respectivamente.

4.2.3 Algoritmos Baseados em Inversões e Entropia para Permutações com Sinais

O Lema 19 mostra que sempre temos uma λ-operação com pontuação de inversões igual a 1. O Teorema 10 apresenta algoritmos gulosos baseados na ideia de alcançar ι ao sempre escolher λ-operações com a maior pontuação de inversões.

Lema 19. *Para qualquer permutação com sinais $\pi \neq \iota$ e $\lambda \geq 2$, sempre existe uma λ-operação σ tal que* $\text{score}_{inv}(\pi, \sigma) = 1$.

Demonstração. Nós dividimos a prova em dois casos, de acordo com os valores de $\text{Inv}(\pi)$.

Se $\text{Inv}(\pi) > 0$, pelo Lema 18, sempre existe uma inversão (π_i, π_{i+1}) e, pelo Lema 4, podemos aplicar uma λ-reversão $\sigma = \rho(i, i+1)$ para obter $\text{Inv}(\pi \cdot \sigma) = \text{Inv}(\pi) - 1$ e $(|E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|) = (|E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|)$. Logo, $\text{score}_{inv}(\pi, \sigma) = 1$.

Se $\text{Inv}(\pi) = 0$, então todos os elementos tem entropia zero. Assim, também temos $|E_{\pi}^{odd^+}| = 0$. Com isso, podemos aplicar uma λ-reversão unitária σ sobre cada elemento negativo para obter $|E_{\pi\cdot\sigma}^{even^-}| = |E_{\pi}^{even^-}| - 1$ ao mesmo tempo que mantemos $\text{Inv}(\pi \cdot \sigma) = |E_{\pi\cdot\sigma}^{odd^+}| = 0$. Logo, $\text{score}_{inv}(\pi, \sigma) = 1$. □ □

Teorema 10. *Os problemas de Ordenação de Permutações por λ-Reversões e por λ-Reversões e λ-Transposições tem algoritmo de $(\frac{\lambda(\lambda-1)}{2} + \lambda)$-aproximação.*

Demonstração. Seja $\lambda \geq 2$ um inteiro e seja $\pi \neq \iota$ uma permutação com sinais. Já que a única permutação com $\text{Inv}(\pi) = |E_{\pi}^{even^-}| = |E_{\pi}^{odd^+}| = 0$ é a identidade, podemos projetar um algoritmo guloso que repetidamente escolhe uma λ-operação σ com o maior valor de $\text{score}_{inv}(\pi, \sigma)$ e ele irá eventualmente ordenar a permutação. Pelo Lema 19, temos uma λ-operação σ tal que $\text{score}_{inv}(\pi, \sigma) = 1$ em π. Logo, o número de operações necessárias pelo nosso algoritmo é no máximo $\text{Inv}(\pi) + |E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|$ e seu fator de aproximação segue diretamente do Corolário 3. □ □

Uma vez que $\text{Inv}(\pi) + |E_{\pi}^{even^-}| + |E_{\pi}^{odd^+}|$ é $O(n^2)$, a análise da complexidade de tempo do algoritmo guloso para os problemas abordados é análoga à análise feita para os algoritmos baseados em inversões apresentados na Subseção 4.2.2.

4.3 Resultados Experimentais

Todos os algoritmos apresentados nessa seção foram implementados para a realização de experimentos com o objetivo de analisar a média dos fatores de aproximação obtidos por eles, considerando uma perspectiva prática.

Como entrada para os algoritmos, nós geramos um conjunto com 1000 permutações aleatórias de tamanho 100 com o número máximo de *breakpoints* cada, e consideramos valores de $\lambda = 5, 10, 15, \ldots, 100$.

A ideia principal do experimento foi comparar os fatores de aproximação que provamos com os fatores de aproximação que os algoritmos obtém na prática. Para o cálculo deste fator de aproximação obtido na prática, dividimos o tamanho da sequência de ordenação obtida para cada permutação dada como entrada pelo máximo dos 4 limitantes inferiores

diferentes para as distâncias dos problemas, já que não conseguimos calcular seus valores exatos. Isso foi feito para cada um dos algoritmos implementados.

Três dos limitantes inferiores considerados para os problemas foram os seguintes: (i) limitantes baseados em entropia, apresentados no Lema 9 e no Lema 10; (ii) limitantes baseados em inversões, apresentados no Lema 2 e no Lema 3, e (iii) limitantes baseados em *breakpoints*, apresentados no Lema 1. O quarto limitante vem dos algoritmos da literatura para os problemas em que as operações não possuem tamanhos limitados, que são os algoritmos que utilizamos para implementar os algoritmos de aproximação para grandes valores de λ, descritos na Seção 4.1. Esses limitantes são válidos para λ-operações, como mostra o Lema 7. Os algoritmos da literatura utilizados em nossos experimentos, juntamente com o quarto limitante inferior considerado em cada problema, são descritos a seguir:

1. Ordenação de Permutações com Sinais por Reversões: o algoritmo ótimo para o problema de Ordenação de Permutações com Sinais por Reversões, disponibilizado pelo *GRIMM (Genome Rearrangements In Man and Mouse)* [47] foi utilizado e, para cada permutação, o tamanho da sequência de ordenação dado por ele foi utilizado como limitante inferior para o problema de Ordenação de Permutações com Sinais por λ-Reversões;

2. Ordenação de Permutações sem Sinais por Reversões: cada permutação sem sinais foi transformada em uma permutação com sinais utilizando o algoritmo $(1.4193+\epsilon)$-aproximação, com $\epsilon > 0$, para a decomposição de ciclos do grafo de *breakpoints* [38] e então a sequência ótima de ordenação para a versão de permutações com sinais foi utilizada. Utilizamos como limitante inferior para o problema de Ordenação de Permutações sem Sinais por λ-Reversões o resultado da divisão do tamanho de cada sequência ótima de ordenação por 1.42, assumindo $\epsilon = 0.007$;

3. Ordenação de Permutações por Transposições: o algoritmo 1.5-aproximação dado por Bafna e Pevzner [5] foi implementado. Eles definiram $c_{odd}(\pi)$ como o número de componentes ímpares no grafo de ciclos, uma estrutura apresentada por eles, e deram o limitante inferior $d_t(\pi) \geq \frac{n+1-c_{odd}(\pi)}{2}$ para o problema. Assim, estamos considerando o mesmo como limitante inferior para o problema de Ordenação de Permutações por λ-Transposições;

4. Ordenação de Permutações com Sinais por Reversões e Transposições: o algoritmo 2-aproximação dado por Walter e coautores [49] foi implementado. Os autores deram o limitante inferior $d_{\bar{rt}}(\pi) \geq n + 1 - c_{odd}(\pi)$ para este problema. Assim, estamos considerando o mesmo como limitante inferior para o problema de Ordenação de Permutações por λ-Reversões e λ-Transposições;

5. Ordenação de Permutações sem Sinais por Reversões e Transposições: o algoritmo 3-aproximação dado por Walter e coautores [49] foi implementado. Os autores consideraram o limitante inferior $d_{rt}(\pi) \geq \frac{b(\pi)}{3}$ mostrado no Lema 1. Assim, estamos considerando o mesmo como limitante inferior para o problema de Ordenação de Permutações sem Sinais por λ-Reversões e λ-Transposições.

Os resultados dos nossos experimentos são apresentados nas figuras 4.2 e 4.3.

Apontamos que, para os problemas de Ordenação de Permutações (com ou sem Sinais) por λ-Reversões, o algoritmo para valores grandes de λ teve o melhor fator de aproximação para a maioria dos valores de λ. Para o problema de Ordenação de Permutações por λ-Transposições, o algoritmo para grandes valores de λ teve melhor fator de aproximação médio para os valores de λ pelo menos 60. Diferentemente do esperado, o algoritmo para grandes valores de λ não foi o melhor (em termos de fator de aproximação médio) nos problemas em que ambas as operações foram permitidas. Para estes problemas, o mesmo algoritmo é pior que os algoritmos baseados em entropia e em inversões, em quase todos os valores de λ considerados nos nossos experimentos.

Para valores de λ no máximo 20, os algoritmos para valores pequenos de λ foram melhores que os algoritmos para valores grandes de λ. No problema de Ordenação de Permutações sem Sinais por λ-Reversões, o algoritmo baseado em entropia apresentou fator de aproximação menor que o algoritmo baseado em inversões. Apesar disso, no geral, o fator de aproximação médio de ambos os algoritmos permaneceu similar em todos os problemas abordados.

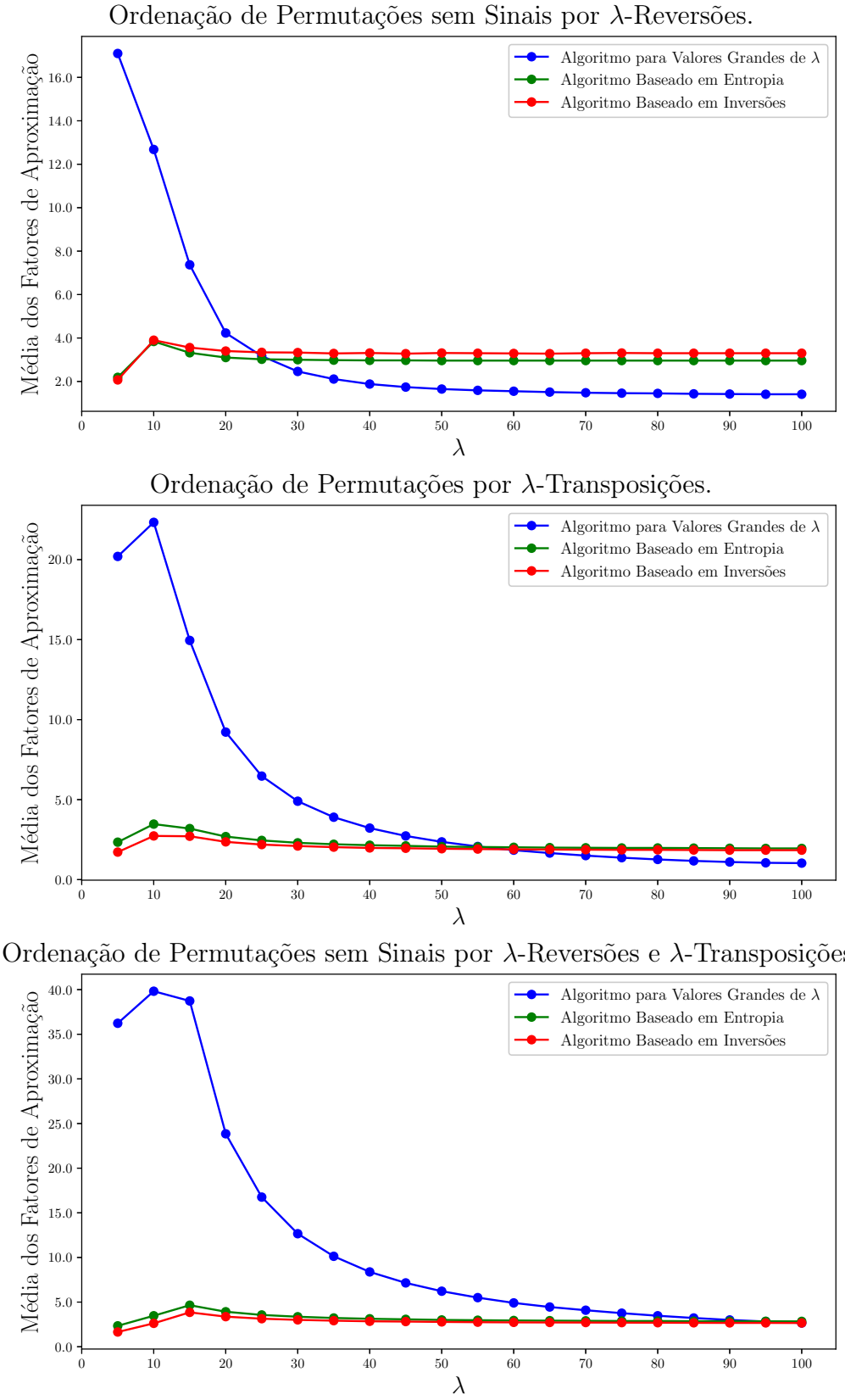


Figura 4.2: Média dos fatores de aproximação dos algoritmos para problemas de Ordenação de Permutações sem Sinais por λ-Operações, utilizando permutações de tamanho 100.

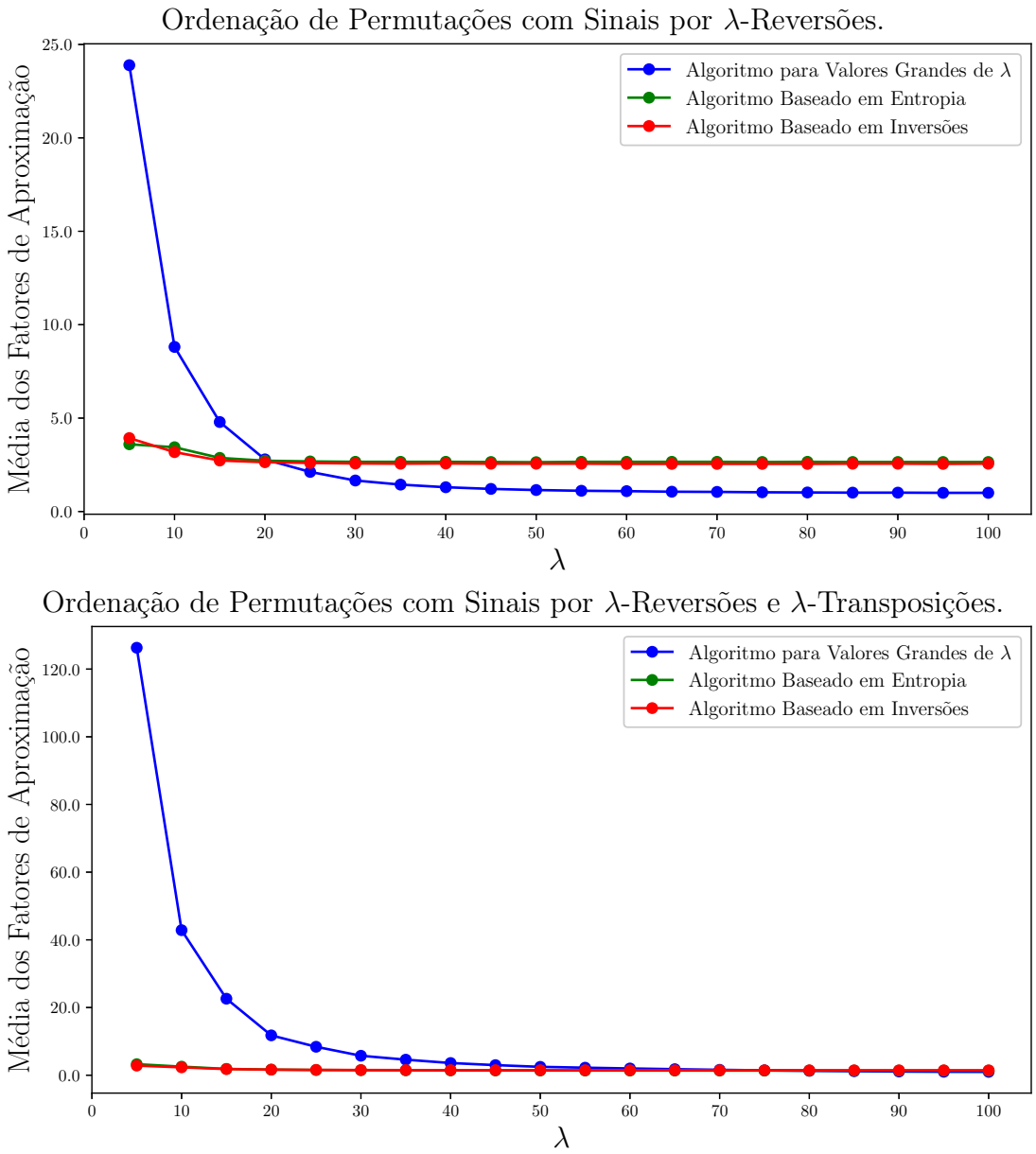


Figura 4.3: Médias dos fatores de aproximação dos algoritmos para problemas de Ordenação de Permutações com Sinais por λ-Operações, utilizando permutações de tamanho 100.

Capítulo 5

Ordenação de λ-Permutações por λ-Operações

Neste capítulo são apresentados algoritmos de aproximação para os 5 problemas de Ordenação de λ-Permutações por λ-Operações que estamos estudando. O capítulo se inicia com a Seção 5.1, que apresenta limitantes sobre o número de λ-permutações existentes. A partir disso, algoritmos de aproximação com fatores $O(\lambda^2)$ são apresentados nas seções 5.2 e 5.3 e algoritmos de aproximação $O(1)$ são apresentados na Seção 5.4. A Seção 5.5 apresenta resultados experimentais de tais algoritmos. A Seção 5.6 apresenta resultados para os diâmetros de ordenação dos problemas abordados. Finalmente, a Seção 5.7 apresenta um algoritmo de aproximação constante para o problema de Ordenação de Permutações por λ-Reversões considerando λ-permutações como entrada.

5.1 Quantas λ-Permutações Existem?

Uma informação interessante sobre este problema é quantas permutações são λ-permutações. Esta seção apresenta limitantes inferiores e superiores sobre o número de λ-permutações com e sem sinais que mostram a existência de um número considerável de λ-permutações.

Lema 20. *Existem pelo menos* $\lambda!^{\lfloor \frac{n}{\lambda} \rfloor}(n \mod \lambda)!$ *λ-permutações sem sinais.*

Demonstração. Uma λ-permutação π de tamanho n pode ser divida em $\lceil \frac{n}{\lambda} \rceil$ segmentos consecutivos de tamanho λ, começando da posição 1 e indo até a posição n. Note que todos os segmentos têm tamanho exatamente λ, exceto, talvez, pelo que termina na posição n e tem tamanho $(n \mod \lambda)$. Em cada segmento de tamanho λ, começando do mais à esquerda e indo até o mais à direita, podemos colocar pelo menos λ elementos, que podem ser arranjados de $\lambda!$ maneiras diferentes. Para o segmento que termina na posição n, temos $(n \mod \lambda)$ elementos remanescentes e então eles podem ser arranjados de $(n \mod \lambda)!$ maneiras diferentes. Já que temos $\lfloor \frac{n}{\lambda} \rfloor$ segmentos de tamanho λ mais um segmento de tamanho $(n \mod \lambda)$, existem pelo menos $\lambda!^{\lfloor \frac{n}{\lambda} \rfloor}(n \mod \lambda)!$ λ-permutações sem sinais. □

Corolário 11. *Existem pelo menos* $2^n\lambda!^{\lfloor \frac{n}{\lambda} \rfloor}(n \mod \lambda)!$ *λ-permutações com sinais.*

Lema 21. *Existem no máximo* $\lambda^{(n-(\lambda-1))}\lambda!$ λ*-permutações com sinais.*

Demonstração. Seja $1 \leq i \leq n$ uma posição qualquer de uma λ-permutação sem sinais π. Seja P_i o conjunto de elementos que podem ser colocados na posição i de acordo com a definição de λ-permutações. Observe que:

$$|P_i| = \begin{cases} \lambda + i - 1, \text{ se } i < \lambda \\ 2\lambda - 1, \text{ se } \lambda \leq i \leq n - (\lambda - 1) \\ \lambda + (n - i), \text{ caso contrário.} \end{cases} \tag{5.1}$$

Então, considere que começamos a construir uma λ-permutação sem sinais colocando elementos um por um, de 1 a n. Note que, na vez de colocar o i-ésimo elemento, existem $|P_i| - |\Psi|$ possíveis escolhas, onde Ψ é o conjunto de elementos que foram colocados nas posições de $\max(1, i - \lambda + 1)$ a $i - 1$. Podemos facilmente ver que $|\Psi| = i - 1$, para $i < \lambda$, e $|\Psi| = \lambda - 1$, para $\lambda \leq i \leq n$. Assim, temos um máximo de $\lambda^{(n-(\lambda-1))}$ maneiras de arranjar os possíveis elementos das posições 1 a $n - (\lambda - 1)$, já que temos no máximo $(\lambda + i - 1) - (i - 1) = \lambda$ e no máximo $(2\lambda - 1) - (\lambda - 1) = \lambda$ elementos para escolher em cada posição $i < \lambda$ e $\lambda \leq i \leq n - (\lambda - 1)$, respectivamente. Como $1 + (n - i)$ elementos podem ser escolhidos para cada posição $i > n - (\lambda - 1)$, temos $\lambda!$ maneiras de arranjar os elementos de tais posições. Logo, existem no máximo $\lambda^{(n-(\lambda-1))}\lambda!$ λ-permutações sem sinais de tamanho n. □

Corolário 12. *Existem no máximo* $2^n\lambda^{(n-(\lambda-1))}\lambda!$ λ*-permutações com sinais.*

5.2 Algoritmos Baseados em Inversões para λ-Permutações sem Sinais

Nesta seção apresentamos algoritmos de aproximação baseados no conceito de inversões para todos os problemas de ordenação de λ-permutações sem sinais que estamos abordando.

O próximo lema mostra que sempre existe uma λ-operação que, quando aplicada em uma λ-permutação, tem outra λ-permutação como resultado.

Lema 22. *Para qualquer* λ*-permutação* $\pi \neq \iota$ *podemos aplicar uma* 2*-reversão ou uma* 2*-transposição para obter uma* λ*-permutação com* $\text{Inv}(\pi) - 1$ *inversões.*

Demonstração. Seja $\pi_j = i$ o menor elemento fora de lugar ($\pi_i \neq i$) em π. Note que $i < j$. Inicialmente, observe que temos uma inversão (π_{j-1}, π_j), já que $\pi_{j-1} > \pi_j$ e $j - 1 < j$. Seja σ uma 2-operação que troca os elementos π_{j-1} e π_j, e seja $\pi' = \pi \cdot \sigma$. É fácil ver que $\text{Inv}(\pi') = \text{Inv}(\pi) - 1$, uma vez que tal inversão foi removida, e note que sempre existe uma λ-reversão ou uma λ-transposição equivalente a σ, pois os elementos são adjacentes e então ambas as operações apenas trocam os dois elementos de lugar.

Observe que, em π', o elemento π_j está mais próximo de sua posição correta, visto que ele foi movido para à esquerda. Consequentemente, mostramos a seguir que π' é uma λ-permutação, ao considerar dois diferentes casos de acordo com os valores de $\pi_{j-1} = \pi'_j$.

Se $\pi_{j-1} \geq j$, então π_{j-1} também está mais próximo de sua posição correta em π'. Caso contrário, $\pi_{j-1} < j$. Note que $\pi_{j-1} > i$. Assim, o elemento π_{j-1} estará, em π', uma posição mais longe de sua posição correta. Então, note que $|j - \pi_j| + 1 = (j - i) + 1 \leq \lambda$, pois π é uma λ-permutação. Também observe que temos $|j - \pi'_j| = j - \pi'_j$, pois $\pi'_j = \pi_{j-1} < j$, e $j - \pi'_j < j - i$, pois $\pi'_j > i$, e, então, $j - i \leq \lambda - 1$. Logo, π' é uma λ-permutação e o resultado segue. □

Um algoritmo de aproximação genérico para os três problemas que estamos considerando nesta seção é apresentado no próximo teorema. Ele recebe um inteiro $\lambda \geq 2$ e uma λ-permutação $\pi \neq \iota$ como entrada. O algoritmo é guloso já que sempre tenta decrementar a maior quantidade de inversões em π. Uma vez que a única permutação sem sinais que não possui inversões é a identidade, ele irá, eventualmente, ordenar π.

Teorema 11. *Existem algoritmos $O(\lambda^2)$-aproximação para o problemas de Ordenação de λ-Permutações sem Sinais por λ-Reversões, por λ-Transposições, e por λ-Reversões e λ-Transposições.*

Demonstração. Seja $\lambda \geq 2$ um inteiro e seja $\pi \neq \iota$ uma λ-permutação. Considere um algoritmo que escolhe a λ-operação σ tal que $\pi \cdot \sigma$ é uma λ-permutação e $\mathrm{Inv}(\pi \cdot \sigma)$ possui o menor valor possível e então aplica tal operação em π. O algoritmo repete o mesmo processo na permutação resultante até alcançar a permutação identidade.

No pior caso, sempre temos uma λ-operação reduzindo o número de inversões em uma unidade, como mostrado no Lema 22. Logo, o número de operações desse algoritmo guloso é no máximo $\mathrm{Inv}(\pi)$, e o fator de aproximação dele segue diretamente do Corolário 2. □

Note que a distância é $O(n^2)$, pois qualquer permutação sem sinais pode ser ordenada com $O(n^2)$ λ-reversões ou λ-transposições. Note também que $\mathrm{Inv}(\pi)$ é $O(n^2)$. Para o problema de Ordenação de Permutações sem Sinais por λ-Reversões, em cada passo o algoritmo considera as $O(\lambda^2)$ possíveis reversões que podem ser escolhidas. Já que a variação no número de inversões causada por uma operação pode ser calculada em tempo $O(\lambda\sqrt{\log \lambda})$ [13], o algoritmo tem complexidade total de tempo $O(n^2\lambda^3\sqrt{\log \lambda})$. Usando a mesma análise, concluímos que os algoritmos envolvendo λ-transposições tem complexidade de tempo total $O(n^2\lambda^4\sqrt{\log \lambda})$.

5.3 Algoritmos Baseados em Inversões e Entropia para λ-Permutações com Sinais

O Lema 23 mostra que sempre existe uma λ-operação σ, para qualquer λ-permutação π, tal que $\mathrm{score}_{inv}(\pi, \sigma) \geq 1$ e $\pi \cdot \sigma$ é uma λ-permutação. Com isto em mente, um algoritmo de aproximação genérico para os dois problemas de ordenação de λ-permutações com sinais que estamos abordando é apresentado no Teorema 12. Ele recebe um inteiro $\lambda \geq 2$ e uma λ-permutação $\pi \neq \iota$ como entrada. O algoritmo é guloso porque sempre escolhe uma λ-operação σ com o maior valor de $\mathrm{score}_{inv}(\pi, \sigma)$ possível. Já que a única permutação com $\mathrm{Inv}(\pi) = |E_\pi^{even^-}| = |E_\pi^{odd^+}| = 0$ é a identidade, ele irá, eventualmente, ordenar π.

Lema 23. *Para qualquer λ-permutação $\pi \neq \iota$ e $\lambda \geq 2$, sempre existe uma λ-operação σ tal que $\pi \cdot \sigma$ é uma λ-permutação e* $\text{score}_{inv}(\pi, \sigma) \geq 1$.

Demonstração. A prova é dividida em dois casos, de acordo com $\text{Inv}(\pi)$.

Primeiro, considere $\text{Inv}(\pi) = 0$. Note que, neste caso, temos apenas elementos com entropia igual a zero em π, implicando que $|E_\pi^{odd^+}| = 0$. Então, ao aplicar uma λ-reversão σ unitária sobre um elemento negativo, obtemos uma λ-permutação $\pi \cdot \sigma$ tal que $|E_{\pi\cdot\sigma}^{even^-}| = |E_\pi^{even^-}| - 1$. Uma vez que tal operação mantém $\text{Inv}(\pi \cdot \sigma) = |E_{\pi\cdot\sigma}^{odd^+}| = 0$, temos $\text{score}(\pi, \sigma) = 1$.

Para $\text{Inv}(\pi) > 0$, o Lema 22 mostra como decrementar uma inversão ao aplicar uma 2-reversão σ e, já que o Lema 4 mostra que $(|E_\pi^{even^-}| + |E_\pi^{odd^+}|) = (|E_{\pi\cdot\sigma}^{even^-}| + |E_{\pi\cdot\sigma}^{odd^+}|)$, temos $\text{score}(\pi, \sigma) = 1$. □

Teorema 12. *Existem algoritmos $(\frac{\lambda(\lambda-1)}{2} + \lambda)$-aproximação para os problemas de Ordenação de λ-Permutações com Sinais por λ-Reversões e por λ-Reversões e λ-Transposições.*

Demonstração. Seja $\lambda \geq 2$ um inteiro e seja $\pi \neq \iota$ uma λ-permutação. Considere um algoritmo que escolhe uma λ-operação σ tal que $\pi \cdot \sigma$ é uma λ-permutação e $\text{score}(\pi \cdot \sigma)$ tem o maior valor possível e então a aplica em π. O algoritmo repete o mesmo processo na permutação resultado até que a permutação identidade seja alcançada.

No pior caso, sempre temos uma λ-reversão com pontuação de inversões igual a 1, como mostrado no Lema 23. Logo, o número de operações de tal algoritmo guloso é no máximo $\text{Inv}(\pi) + |E_\pi^{even^-}| + |E_\pi^{odd^+}|$, e o fator de aproximação segue imediatamente do Corolário 3. □

Já que $\text{Inv}(\pi)+|E_\pi^{even^-}|+|E_\pi^{odd^+}|$ é $O(n^2)$ e o tempo para calcular a variação no número $|E_\pi^{even^-}| + |E_\pi^{odd^+}|$ é $O(\lambda)$, a análise da complexidade de tempo é análoga aos algoritmos baseados em inversões para λ-permutações sem sinais vistos na Seção 5.2.

5.4 Algoritmos Baseados em Breakpoints

Nesta seção apresentamos algoritmos de aproximação baseados no conceito de *breakpoints* para todos os problemas de ordenação de λ-permutações abordados nesta dissertação.

O próximo lema supõe que o menor elemento fora de lugar está em uma *strip* crescente de uma λ-permutação $\pi \neq \iota$ e então mostra como reduzir o número de *breakpoints* de π movendo tal *strip* para sua posição correta, mas sem utilizar λ-operações. Ele é auxiliar aos lemas 25 e 26, que mostram como fazer isso utilizando uma sequência de λ-operações. O Lema 27 mostra como fazer isso quando o menor elemento fora de lugar está em uma *strip* decrescente.

Lema 24. *Seja π uma λ-permutação. Seja $|\pi_j| = i$ o menor elemento fora de lugar em π. Suponha que π_j está em uma strip crescente $S = (\pi_j \ \ldots \ \pi_k)$. Então $\pi \cdot \tau(i, j, k+1)$ é uma λ-permutação, $b(\pi \cdot \tau(i, j, k+1)) \leq b(\pi) - 1$ e $(k + 1 - i) \leq 2(\lambda - 1)$.*

Demonstração. Seja $R = (\pi_i \ldots \pi_{j-1})$ o segmento de elementos em π que serão transpostos com S. Observe que o valor absoluto de qualquer elemento em R é maior que qualquer

elemento em S. então $\pi \cdot \tau(i, j, k+1)$ é uma λ-permutação, uma vez que elementos maiores (em seus valores absolutos) são movidos para à direita, enquanto elementos menores são movidos para à esquerda. Também observe que em π temos 3 *breakpoints* (π_{i-1}, π_i), (π_j, π_{j+1}) e (π_{k-1}, π_k), onde o primeiro é porque $\pi_{i-1} = \pi_j - 1 = i - 1$ e $|\pi_i| > i = \pi_j$ e o segundo e terceiro são porque o começo e o fim da *strip* estão nas posições j e k, respectivamente. A transposição $\tau(i, j, k+1)$ move os elementos de S para suas posições corretas ao transpor eles com os elementos de R, assim removendo pelo menos o *breakpoint* (π_{i-1}, π_i). Já que uma transposição pode adicionar no máximo 3 *breakpoints*, mas já temos todos eles e removemos pelo menos (π_{i-1}, π_i), temos $b(\pi \cdot \tau(i, j, k+1)) \leq b(\pi) - 1$.

Pelo Lema 2, temos $|S| \leq \lambda - 1$, assim $k + 1 - j \leq \lambda - 1$. Já que π é uma λ-permutação, temos $||\pi_j| - j| \leq \lambda - 1$, e, por construção, $\pi_j = i$, assim $|i - j| + 1 = j - i + 1 \leq \lambda - 1$. Logo, $k + 1 - i \leq 2(\lambda - 1)$. □

Lema 25. *Seja π uma λ-permutação. Seja $|\pi_j| = i$ o menor elemento fora de lugar em π. Suponha que π somente tem strips crescentes e que π_j está em uma strip $S = (\pi_j \ \ldots \ \pi_k)$. Sempre é possível obter uma λ-permutação $\pi \cdot \tau(i, j, k+1)$ com no máximo $b(\pi) - 1$ breakpoints ao aplicar no máximo $5 + \lceil (\lambda - 1)/2 \rceil$ λ-reversões tais que todas permutações intermediárias são λ-permutações.*

Demonstração. Seja $R = (\pi_i \ \ldots \ \pi_{j-1})$ o segmento que será movido para a direita em $\tau(i, j, k+1)$. Note que $|S| \leq \lambda - 1$, pelo Lema 2, e $|R| \leq \lambda - 1$, porque π é uma λ-permutação.

A ideia é mover elementos de S para suas posições corretas ao aplicar no máximo duas sequências de pares de λ-reversões, onde cada uma coloca no máximo $\lfloor \lambda/2 \rfloor$ elementos de S em suas posições corretas por vez.

Na primeira sequência de λ-reversões, existem duas possibilidades. Se $|S| \leq \lfloor \lambda/2 \rfloor$, então a primeira operação de cada par reverte $|S|$ elementos contidos em S e no máximo $|S|$ elementos contidos em R. Se $|S| > \lfloor \lambda/2 \rfloor$, então ela reverte $\lfloor \lambda/2 \rfloor$ elementos contidos em ambos S e R. Em qualquer um dos casos, a segunda operação de cada par reverte novamente os elementos de R que foram afetados pela primeira operação e, com isso, π volta a ter somente *strip* crescentes (com exceção dos elementos de S que foram afetados pela primeira operação).

Depois da sequência ser aplicada, temos no máximo $\lfloor \lambda/2 \rfloor$ elementos de S das posições i a $i + \min(\lfloor \lambda/2 \rfloor, |S|)$, e, talvez, eles estejam em uma *strip* decrescente. Se este é o caso, então uma λ-reversão extra tem que ser aplicada para colocar esses elementos em suas posições corretas, revertendo tal *strip* decrescente.

A segunda sequência de λ-reversões coloca no máximo $\lfloor \lambda/2 \rfloor$ elementos remanescentes de S em suas posições corretas, seguindo a mesma ideia, e, também, talvez uma λ-reversão extra seja necessária após esta sequência ser aplicada. Note que, se não existem elementos remanescentes (no caso em que $|S| \leq \lfloor \lambda/2 \rfloor$), a segunda sequência não é necessária.

A maior quantidade de operações necessárias no processo descrito acima acontece quando temos exatamente $\lfloor \lambda/2 \rfloor + 1$ elementos em S. Já que $|S| > \lfloor \lambda/2 \rfloor$, o processo começa movendo os primeiros $\lfloor \lambda/2 \rfloor$ elementos de S para suas posições corretas. Observe que, para cada par de reversões aplicado na primeira sequência, uma delas move $\lfloor \lambda/2 \rfloor$ elementos $\lfloor \lambda/2 \rfloor$ posições à esquerda (exceto, talvez, pelo último par), e então a segunda

operação reverte novamente os elementos de R afetados pela primeira reversão. Além disso, pela definição de λ-permutações, os elementos de S estão a no máximo $\lambda - 1$ de distância de suas posições corretas e logo no máximo 2 pares de reversões são necessários para colocá-los a partir da posição i. Como dito antes, talvez tais elementos de S terminam em ordem reversa e, nesse caso, uma reversão a mais é necessária. Assim, no máximo 5 operações podem ter sido aplicadas nesta primeira etapa.

Para mover o elemento remanescente de S para sua posição correta, todas as λ-reversões da segunda sequência terão tamanho 2 (note que, neste caso, não necessitamos da segunda operação de cada par), o que significa que tal elemento será movido somente 1 posição à esquerda por operação, acrescentando uma quantidade extra de $\lambda - 1$ λ-reversões. Logo, o número de λ-reversões para mover S para sua posição correta é no máximo $5 + \lambda - 1$.

Agora temos que mostrar que depois que cada operação é aplicada, temos uma λ-permutação como resultado e, depois que a última λ-permutação é aplicada, temos $\pi \cdot \tau(i, j, k+1)$. Observe que qualquer elemento em R é maior do que qualquer elemento em S. Então, uma vez que a primeira operação de cada par move os elementos de R para a direita e os elementos de S para à esquerda, todos os elementos afetados estão mais próximos de suas posições corretas, resultando em uma λ-permutação. A segunda operação de cada par reverte os elementos de R para ordem crescente novamente, então ela também resulta em uma λ-permutação. Depois de ambas as sequências de λ-reversões serem aplicadas, todos elementos de S estão nas posições de i a $i + k - j$ e todos elementos de R estão nas posições de $i + k - 1$ a k, resultando em $\pi \cdot \tau(i, j, k + 1)$, que é uma λ-permutação com pelo menos um *breakpoint* a menos que π, como mostrado no Lema 24. □

Como um exemplo para o Lema 25, considere uma 4-permutação com sinais $\pi = (+4\ +5\ +6\ +1\ +2\ +3)$. Seguindo a prova do lema, temos $S = (+1\ +2\ +3)$ e $R = (+4\ +5\ +6)$. As seguintes reversões (sublinhadas em cada permutação) são utilizadas para posicionar S:

$$\begin{aligned}
\pi &= (+4\ \underline{+5\ +6\ +1\ +2}\ +3) \cdot \rho(2,5)\\
\pi^1 &= (+4\ -2\ -1\ \underline{-6\ -5}\ +3) \cdot \rho(4,5)\\
\pi^2 &= (\underline{+4\ -2\ -1}\ +5\ +6\ +3) \cdot \rho(1,3)\\
\pi^3 &= (+1\ +2\ \underline{-4}\ +5\ +6\ +3) \cdot \rho(3,3)\\
\pi^4 &= (+1\ +2\ +4\ \underline{+5\ +6\ +3}) \cdot \rho(4,6)\\
\pi^5 &= (+1\ +2\ +4\ -3\ \underline{-6\ -5}) \cdot \rho(5,6)\\
\pi^6 &= (+1\ +2\ \underline{+4\ -3}\ +5\ +6) \cdot \rho(3,4)\\
\pi^7 &= (+1\ +2\ +3\ \underline{-4}\ +5\ +6) \cdot \rho(4,4)\\
\pi^8 &= (+1\ +2\ +3\ +4\ +5\ +6)
\end{aligned}$$

No exemplo, a reversão $\rho(3, 3)$ é a última aplicada na primeira sequência da prova, e a reversão $\rho(4, 4)$ é a última aplicada na segunda sequência da prova.

Lema 26. *Seja π uma λ-permutação. Seja $|\pi_j| = i$ o menor elemento fora de lugar em π. Suponha que π_j está em uma strip crescente $S = (\pi_j \ \ldots \ \pi_k)$. É sempre possível obter*

uma λ-permutação $\pi \cdot \tau(i, j, k+1)$ com no máximo $b(\pi) - 1$ breakpoints ao aplicar no máximo 4 λ-transposições tais que todas permutações intermediárias são λ-permutações.

Demonstração. Seja $R = (\pi_i \ \ldots \ \pi_{j-1})$ o segmento que será movido para a direita em $\tau(i, j, k+1)$. Note que $|S| \leq \lambda - 1$, pelo Lema 2, e $|R| \leq \lambda - 1$, porque π é uma λ-permutação.

A ideia é aplicar uma sequência com no máximo 4 λ-transposições que dividem ambos segmentos $R = (\pi_i \ \ldots \ \pi_{j-1})$ e $S = (\pi_j \ \ldots \ \pi_k)$ em no máximo duas partes cada, onde cada parte tem no máximo $\lfloor \lambda/2 \rfloor$ elementos, e então trocar cada parte de S no máximo duas vezes (e pelo menos uma vez), com as (possíveis) duas partes de R. Por exemplo, no caso em que temos exatamente $\lambda - 1$ elementos em cada segmento S e R, a sequência de operações será $\tau(i + \lfloor \lambda/2 \rfloor, j, j + \lfloor \lambda/2 \rfloor), \tau(i, i + \lfloor \lambda/2 \rfloor, j), \tau(j, j + \lfloor \lambda/2 \rfloor, k + 1), \tau(i + \lfloor \lambda/2 \rfloor, j, j + \lfloor \lambda/2 \rfloor)$.

Agora temos que mostrar que depois que cada uma das 4 operações é aplicada, temos uma λ-permutação como resultado. Observe que o valor absoluto de qualquer elemento em R é maior que qualquer elemento em S. Já que cada λ-transposição coloca os elementos de S mais próximos de suas posições corretas ao transpor eles com elementos maiores (em seus valores absolutos) de R, temos uma λ-permutação para cada λ-operação aplicada. Depois que todas λ-transposições são aplicadas, os elementos de S estão nas posições de i a $i + k - j$ e os elementos de R estão nas posições de $i + k - j + 1$ a k, resultando em $\pi \cdot \tau(i, j, k+1)$, que é uma λ-permutação com pelo menos $b(\pi) - 1$ *breakpoints*, como mostrado no Lema 24. □

Lema 27. *Seja π uma λ-permutação. Seja $|\pi_k| = i$ o menor elemento fora de lugar em π. Suponha que π_k está em uma strip decrescente $S = (\pi_j \ \ldots \ \pi_k)$. Sempre é possível obter uma λ-permutação com no máximo $b(\pi) - 1$ breakpoints ao aplicar no máximo uma λ-transposição e uma λ-reversão.*

Demonstração. Quando $j = i$, uma reversão $\rho(j, k)$ coloca os elementos de S em suas posições corretas. É fácil ver que $\pi \cdot \rho(j, k)$ é uma λ-permutação e, uma vez que $|S| = k - j + 1 \leq \lambda - 1$ pelo Lema 2, temos que $\rho(j, k)$ é uma λ-reversão.

Agora assuma $j > i$. Note que, neste caso, temos os 3 *breakpoints* (π_{i-1}, π_i), (π_j, π_{j+1}), e (π_{k-1}, π_k), onde o primeiro é porque $\pi_{i-1} = |\pi_k| - 1 = i - 1$ e $|\pi_i| > i = |\pi_k|$ e o segundo e terceiro são porque o começo e o fim da *strip* estão nas posições k e j, respectivamente. Assim, podemos aplicar a λ-transposição $\tau(i, j, k+1)$ seguida pela λ-reversão $\rho(i, i + (k - j))$ e então obtemos $b(\pi \cdot \tau(i, j, k+1) \cdot \rho(i, i + (k - j))) \leq b(\pi) - 1$, uma vez que uma λ-transposição pode adicionar no máximo 3 *breakpoints* mas já tínhamos (π_{i-1}, π_i), (π_{j-1}, π_j), e (π_k, π_{k+1}), e a segunda λ-reversão pode adicionar no máximo dois *breakpoints*, mas já tínhamos (π_{i-1}, π_j) e (π_k, π_i) e nós removemos o primeiro deles, visto que todos os elementos de S estão em suas posições corretas em $\pi \cdot \tau(i, j, k+1) \cdot \rho(i, i + (k - j))$.

Agora temos que mostrar que depois que cada operação é aplicada, temos uma λ-permutação como resultado. Seja $R = (\pi_i \ \ldots \ \pi_{j-1})$ o segmento de elementos que devem ser movidos para colocar S em sua posição correta. Observe que o valor absoluto de qualquer elemento em R é maior que qualquer elemento em S. A primeira operação, uma λ-transposição, transpõe S somente com elementos maiores (em seus valores absolutos)

e assim o resultado é uma λ-permutação. A segunda operação, uma λ-reversão, apenas reverte uma *strip* decrescente para colocar os elementos de S em suas posições corretas, assim isto também resulta em uma λ-permutação. Consequentemente, temos como resultado uma λ-permutação com pelo menos um *breakpoint* a menos. □

Os próximos teoremas descrevem algoritmos de aproximação para os problemas de ordenação de λ-permutações abordados no capítulo. O Lema 28 é auxiliar ao Teorema 13. Os algoritmos recebem um inteiro $\lambda \geq 2$ e uma λ-permutação $\pi \neq \iota$ como entrada. O objetivo é decrementar pelo menos uma unidade no número de *breakpoints* em π ao mover elementos para suas posições corretas, aplicando o Lema 26 e o Lema 27. Já que a única permutação sem nenhum *breakpoint* é a identidade, os algoritmos irão, eventualmente, ordenar π.

Lema 28. *Seja π uma λ-permutação. Seja $S = (j \dots i)$ uma strip decrescente em π (assim $|i| < |j|$). Seja $\pi' = \pi \cdot \rho(\pi^{-1}_{|j|}, \pi^{-1}_{|i|})$ a permutação resultante depois de reverter S em π. Então, π' é uma λ-permutação.*

Demonstração. Primeiro note que o elemento π_j está à direita do elemento π_i. Nós mostramos que o lema segue ao considerar quatro casos, de acordo com as posições dos elementos i e j em relação com os elementos $\pi_{|i|}$ e $\pi_{|j|}$.

Caso (i), $|i| < |j| < \pi^{-1}_{|j|} < \pi^{-1}_{|i|}$: note que ambos $\pi_{|i|}$ e $\pi_{|j|}$ estão à esquerda de S. Então, depois de reverter S, o elemento i está mais próximo de sua posição correta, enquanto o elemento j está mais longe de sua posição correta. Apesar disto, a distância entre $\pi_{|j|}$ e j em π' é menor que a distância entre $\pi_{|i|}$ e i em π, e então se π é uma λ-permutação, π' também é uma λ-permutação.

Caso (ii), $|i| < \pi^{-1}_{|j|} \leq |j| < \pi^{-1}_{|i|}$: note que $\pi_{|i|}$ está à esquerda de S e $\pi_{|j|}$ está em S. Então, depois de reverter S, o elemento i está mais próximo de sua posição correta e a distância de j para sua posição correta ainda será menor que λ, pois o tamanho de S é no máximo λ, como o Lema 2 mostra.

Caso (iii), $\pi^{-1}_{|j|} \leq |i| < \pi^{-1}_{|i|} \leq |j|$: similar ao caso (ii).

Caso (iv), $\pi^{-1}_{|j|} < \pi^{-1}_{|i|} \leq |i| < |j|$: similar ao caso (i). □

Teorema 13. *Os problemas de Ordenação de λ-Permutações com e sem Sinais por λ-Reversões têm algoritmos $(10 + 2\lambda)$-aproximação*

Demonstração. Seja $\lambda \geq 2$ um inteiro e $\pi \neq \iota$ uma λ-permutação. Considere um algoritmo que primeiro aplica uma λ-reversão sobre cada *strip* decrescente de π para obter uma λ-permutação que possui apenas *strip* crescentes. O Lema 28, garante que todas as permutações intermediárias geradas por tais λ-reversões são λ-permutações.

Então, o algoritmo irá repetidamente pegar o menor elemento fora de lugar e mover a *strip* crescente que o contém para sua posição correta, obtendo uma λ-permutação com pelo menos um *breakpoint* a menos, até alcançar a permutação identidade.

Como mostrado no Lema 25, no máximo $5 + \lambda - 1$ λ-reversões são necessárias para mover cada *strip* para sua posição correta. Já que, talvez, uma λ-reversão extra poderá ter que ser aplicada no início do algoritmo para transformar todas as *strip* em *strip* crescentes,

temos que no máximo $6+\lambda-1$ λ-reversões podem ser aplicadas para remover pelo menos um *breakpoint*. Logo, o número de operações do nosso algoritmo é no máximo

$$(6+\lambda-1)b(\pi) \leq 2(6+\lambda-1)d_{\beta}^{\lambda*}(\pi) = (10+2\lambda)d_{\beta}^{\lambda*}(\pi),$$

para $\beta \in \{r, \bar{r}\}$, e a inequação segue do Lema 1. □

Teorema 14. *O problema de Ordenação de λ-Permutações por λ-Transposições tem um algoritmo* 12*-aproximação.*

Demonstração. Seja $\lambda \geq 2$ um inteiro e seja $\pi \neq \iota$ uma λ-permutação. O algoritmo irá repetidamente pegar o menor elemento fora de lugar e mover a *strip* crescente que o contém para sua posição correta, obtendo uma λ-permutação com pelo menos um *breakpoint* a menos, até alcançar a permutação identidade.

Como mostrado no Lema 26, no máximo 4 λ-transposições são necessárias para mover cada *strip* para sua posição correta. Então, no pior caso, removemos 1 *breakpoint* em todas sequências de 4 λ-transposições aplicadas. Juntamente com isso e o Lema 1, concluímos que o número de operações do nosso algoritmo é no máximo $4b(\pi) \leq 12d_{t}^{\lambda*}(\pi)$. □

Teorema 15. *Os problemas de Ordenação de λ-Permutações com e sem Sinais por λ-Reversões e λ-Transposições têm algoritmos* 12*-aproximação.*

Demonstração. Seja $\lambda \geq 2$ um inteiro e seja $\pi \neq \iota$ uma λ-permutação. Seja $\pi_j = i$ o menor elemento fora de lugar em π.

Temos dois casos para considerar: quando a *strip* que contém π_j é crescente ou não. Em ambos os casos, podemos remover pelo menos o *breakpoint* (π_{i-1}, π_i) de π sem adicionar outros ao aplicar no máximo 4 λ-transposições (se a *strip* é crescente) ou no máximo 2 λ-operações (se a *strip* é decrescente), como mostrado nos lemas 26 e 27, respectivamente.

Então, considerando ambos os casos descritos, o algoritmo irá repetidamente pegar o menor elemento fora de lugar e mover a *strip* que o contém para sua posição correta, removendo pelo menos um *breakpoint* por vez, até alcançar a permutação identidade.

Note que, no pior caso, removemos 1 *breakpoint* em todas sequências de 4 λ-transposições aplicadas e então o resultado é análogo ao Teorema 14. □

Note que $b(\pi)$ é $O(n)$ e a complexidade de tempo para encontrar a *strip* com o menor elemento fora de lugar em π é $O(n)$. Assim, concluímos que as complexidades de tempo para os algoritmos $O(1)$-aproximação e $O(\lambda)$-aproximação são $O(n(n+\lambda))$ e $O(n(n+\lambda^2))$, respectivamente.

5.5 Resultados Experimentais

Todos os algoritmos apresentados neste capítulo foram implementados para a realização de experimentos com o objetivo de comparar o fator de aproximação médio obtido por eles em uma perspectiva prática. Como entrada para nossos algoritmos consideramos um total de 1000 λ-permutações com e sem sinais de tamanho 100 e valores de $\lambda = 5, 10, 15, \ldots, 100$. As λ-permutações utilizadas foram geradas de duas maneiras diferentes: (i) totalmente

aleatórias, e (ii) aplicando 20 λ-operações aleatórias (de acordo com o modelo de rearranjo de cada problema) sobre a identidade. Reforçamos que todas as permutações geradas são λ-permutações em ambos os casos, incluindo as permutações intermediárias para o caso (ii). Para (i), as λ-permutações tendem a ter uma grande quantidade de *breakpoints*, e para (ii) elas tendem a uma grande quantidade de *strips* com mais de um elemento e, além disso, sabemos que a distância é no máximo 20.

Então, nós comparamos os resultados de acordo com os fatores de aproximação médio e máximo obtidos para todas as permutações. Para a realização do cálculo do fator, dividimos o tamanho da sequência de ordenação obtida por cada permutação dada com entrada pelo máximo valor entre os limitantes inferiores apresentados no Lema 1 e nos corolários 2 e 3.

Os resultados experimentais para λ-permutações totalmente aleatórias são apresentados nas figuras 5.1 e 5.2, e para λ-permutações geradas a partir da identidade nas figuras 5.3 e 5.4.

Considerando os resultados para λ-permutações totalmente aleatórias, nós observamos que os fatores de aproximação máximos para os problemas de ordenação de λ-permutações sem sinais foram 5.38 e 6.76 para λ-reversões, 2.91 e 3.15 para λ-transposições, e 3.00 e 3.18 para quando ambas as λ-operações são permitidas, considerando os algoritmos baseados em *breakpoints* e em inversões, respectivamente. Para os problemas de ordenação de λ-permutações com sinais, os fatores de aproximação máximos foram 7.78 e 7.54 para λ-reversões e 4.74 e 5.21 para quando ambas as λ-operações são permitidas, considerando os algoritmos baseados em *breakpoints* e em inversões, respectivamente.

Uma outra observação é que, para permutações totalmente aleatórias, o fator de aproximação médio dos algoritmos baseados em inversões e em *breakpoints* foram similares (exceto para o problema de Ordenação de λ-Permutações com Sinais por λ-Reversões), mesmo com a relevante diferença entre seus fatores de aproximação teóricos.

Considerando os resultados para λ-permutações aleatórias geradas a partir da identidade, nós observamos que os fatores de aproximação máximos para os problemas de ordenação de λ-permutações sem sinais foram 6.08 e 18.5 para λ-reversões, 2.83 e 2.64 para λ-transposições, e 4.33 e 3.58 para quando ambas as λ-operações são permitidas, considerando os algoritmos baseados em *breakpoints* e em inversões, respectivamente. Para os problemas de ordenação de λ-permutações com sinais, os fatores de aproximação máximos foram 7.13 e 22.85 para λ-reversões e 5.00 e 11.56 para quando ambas as λ-operações são permitidas, considerando os algoritmos baseados em *breakpoints* e em inversões, respectivamente.

Uma outra observação é que, para este segundo tipo de experimentos com λ-permutações, os fatores de aproximação máximos dos algoritmos baseados em inversões para os problemas de ordenação de λ-permutações com sinais e para o problema de Ordenação de λ-Permutações Sem Sinais por λ-Reversões foram consideravelmente maiores quando comparados com seus fatores de aproximação médios. Apesar disso, os fatores de aproximação médios e máximos dados pelos algoritmos baseados em inversões foram melhores para os dois problemas de permutações sem sinais que permitem λ-transposições. Para ambos os problemas que permitem apenas λ-reversões, considerando λ-permutações com e sem sinais, os algoritmos baseados em *breakpoints* tiveram fatores de aproxima-

ção médios e máximos melhores. Para o problema de Ordenação de λ-Permutações com Sinais por λ-Reversões e λ-Transposições, ambos os algoritmos tiveram fatores de aproximação médios similares, mas os algoritmos baseados em *breakpoints* tiveram fatores de aproximação máximos melhores e mais constantes.

5.6 Diâmetro

Esta seção apresenta resultados sobre os diâmetros dos problemas de ordenação de λ-permutações que estamos considerando. Seja n o número elementos de uma λ-permutação. As tabelas 5.1, 5.2, 5.3, 5.4 e 5.5 apresentam os valores dos diâmetros para todos os problemas considerados, com λ-permutações sem sinais e $n \in \{3, 4, \ldots, 10\}$, λ-permutações com sinais e $n \in \{3, 4, \ldots, 9\}$, e $\lambda \in \{3, 4, \ldots, n\}$.

n	Tamanho Máximo de uma Operação (λ)							
-	2	3	4	5	6	7	8	9
3	1	2	-	-	-	-	-	-
4	2	3	3	-	-	-	-	-
5	2	4	4	4	-	-	-	-
6	3	5	5	5	5	-	-	-
7	3	6	6	6	6	6	-	-
8	4	7	7	7	7	7	7	-
9	4	8	8	8	8	8	8	8

Tabela 5.1: Diâmetros de Ordenação de λ-Permutações sem Sinais por λ-Reversões.

n	Tamanho Máximo de uma Operação (λ)							
-	2	3	4	5	6	7	8	9
3	1	2	-	-	-	-	-	-
4	2	2	3	-	-	-	-	-
5	2	3	3	3	-	-	-	-
6	3	4	4	4	4	-	-	-
7	3	4	5	5	5	4	-	-
8	4	5	6	5	5	5	5	-
9	4	6	6	6	6	6	6	5

Tabela 5.2: Diâmetros de Ordenação de λ-Permutações por λ-Transposições.

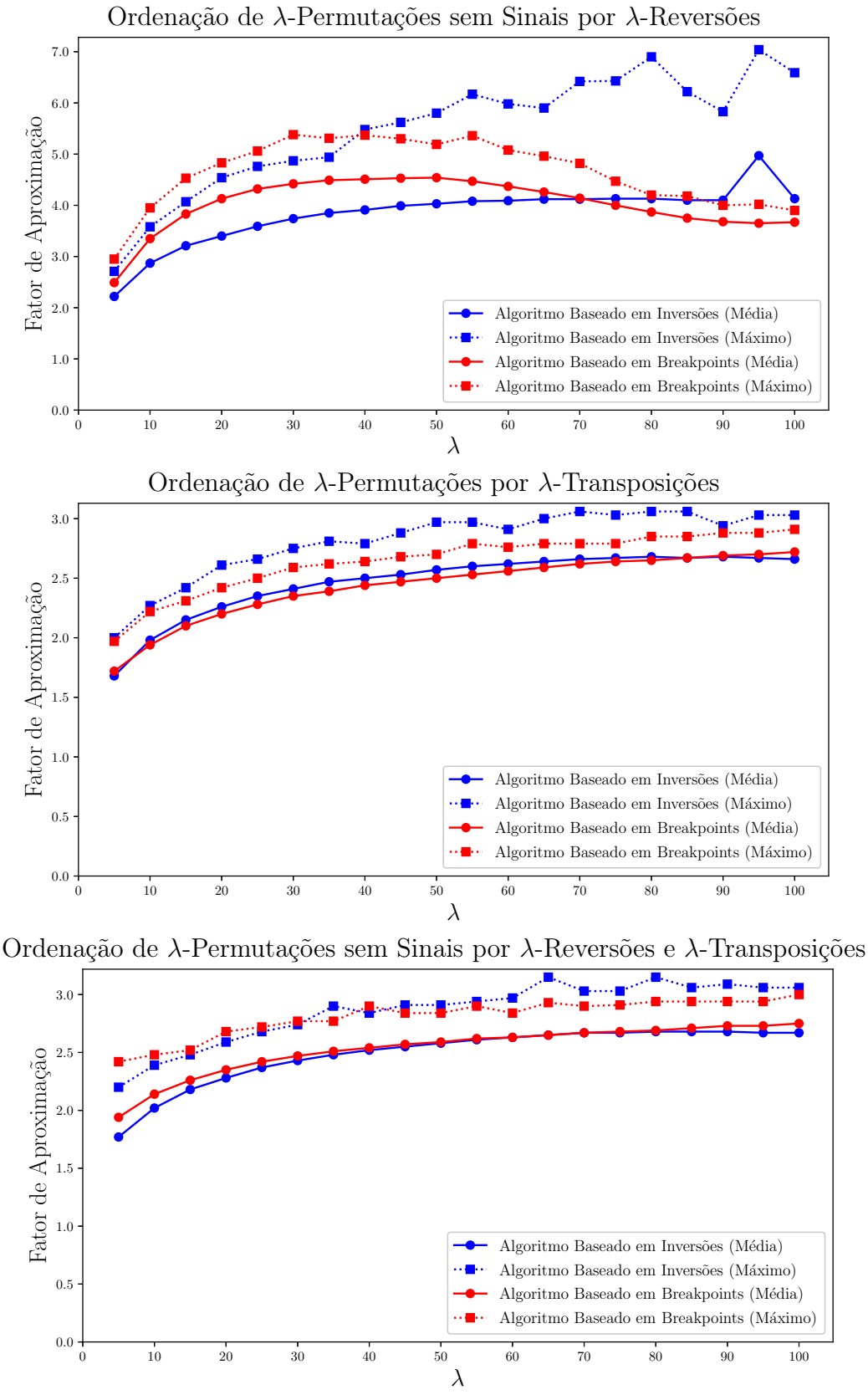


Figura 5.1: Médias dos fatores de aproximação e fatores de aproximação máximos dos algoritmos para os problemas de Ordenação de λ-Permutações sem Sinais por λ-Operações, com λ-permutações de tamanho 100 geradas de forma totalmente aleatória.

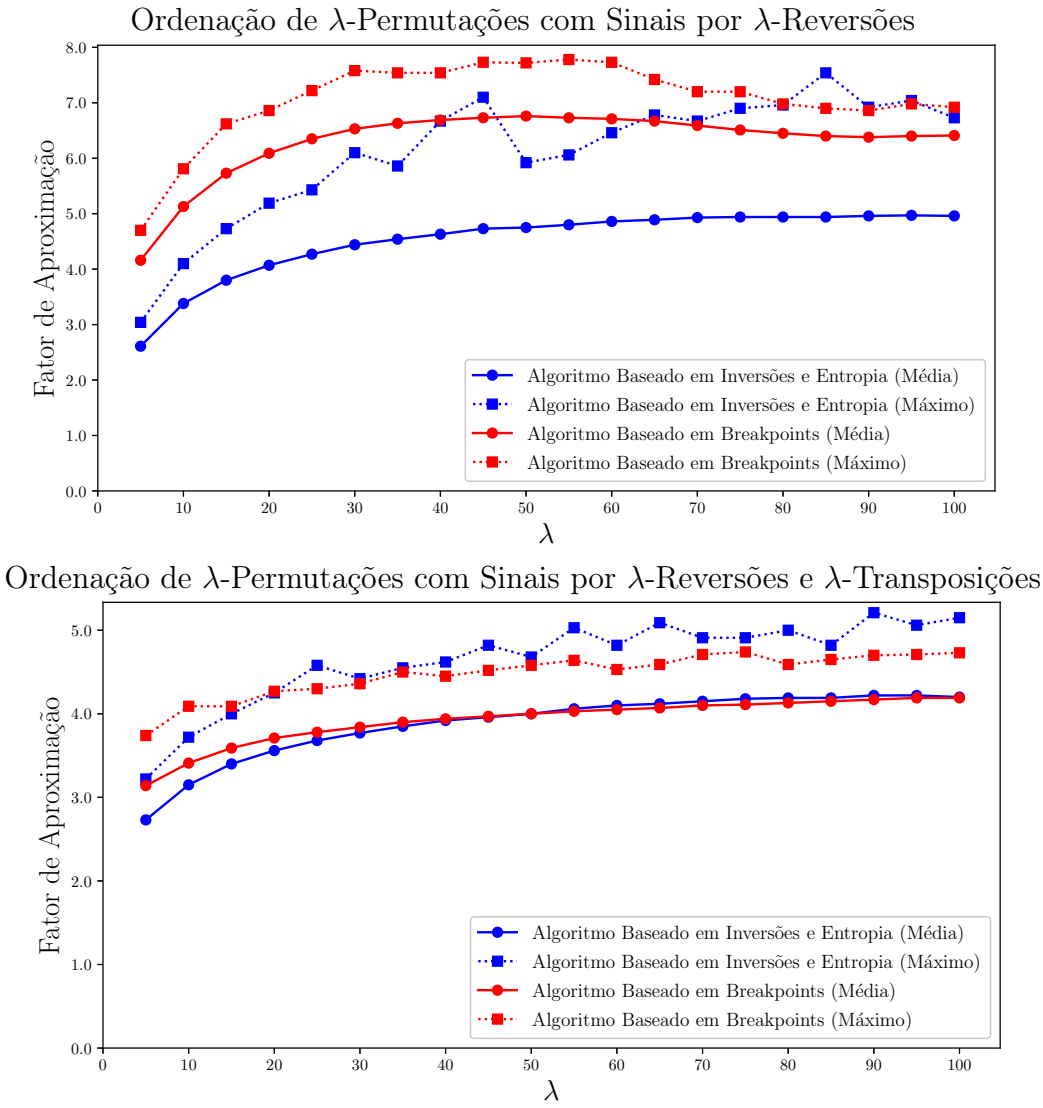


Figura 5.2: Médias dos fatores de aproximação e fatores de aproximação máximos dos algoritmos para os problemas de Ordenação de λ-Permutações com Sinais por λ-Operações, com λ-permutações de tamanho 100 geradas de forma totalmente aleatória.

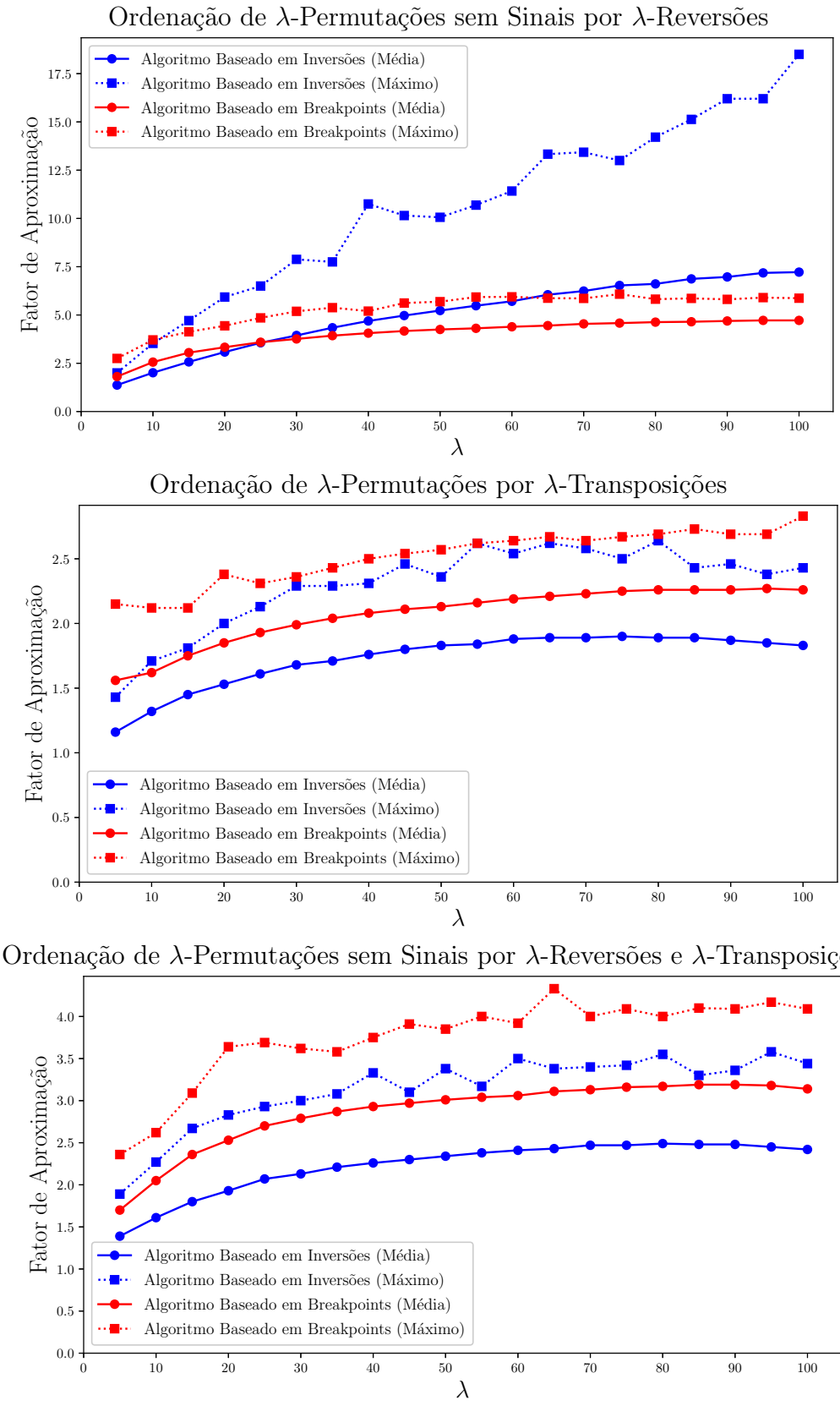


Figura 5.3: Médias dos fatores de aproximação e fatores de aproximação máximos dos algoritmos para o problema de Ordenação de λ-Permutações sem Sinais por λ-Operações, com λ-permutações aleatórias de tamanho 100 geradas a partir de ι.

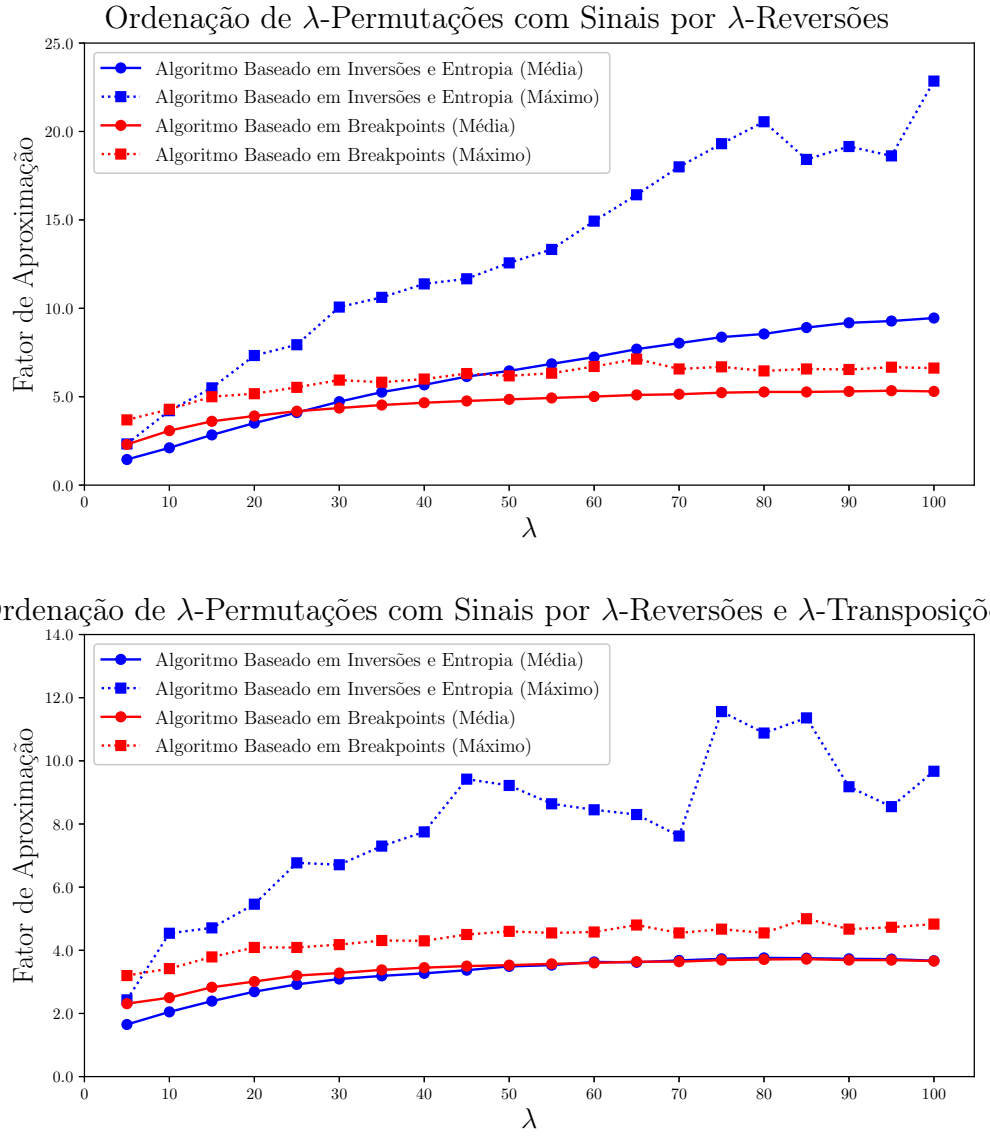


Figura 5.4: Médias dos fatores de aproximação e fatores de aproximação máximos dos algoritmos para o problema de Ordenação de λ-Permutações com Sinais por λ-Operações, com λ-permutações aleatórias de tamanho 100 geradas a partir de ι.

n	Tamanho Máximo de uma Operação (λ)							
-	2	3	4	5	6	7	8	9
3	1	1	-	-	-	-	-	-
4	2	2	2	-	-	-	-	-
5	2	2	3	3	-	-	-	-
6	3	3	3	3	3	-	-	-
7	3	3	4	4	4	4	-	-
8	4	4	5	4	4	4	4	-
9	4	4	5	5	5	5	5	5

Tabela 5.3: Diâmetros de Ordenação de λ-Permutações sem Sinais por λ-Reversões e λ-Transposições.

n	Tamanho Máximo de uma Operação (λ)							
-	2	3	4	5	6	7	8	9
3	4	3	-	-	-	-	-	-
4	6	5	5	-	-	-	-	-
5	7	6	6	6	-	-	-	-
6	9	7	7	7	7	-	-	-
7	10	8	9	8	8	8	-	-
8	12	10	10	9	9	9	9	-
9	13	11	11	11	10	10	10	10

Tabela 5.4: Diâmetros de Ordenação de λ-Permutações com Sinais por λ-Reversões.

n	Tamanho Máximo de uma Operação (λ)							
-	2	3	4	5	6	7	8	9
3	3	3	-	-	-	-	-	-
4	4	4	4	-	-	-	-	-
5	5	5	5	4	-	-	-	-
6	6	6	5	5	5	-	-	-
7	7	7	6	6	6	6	-	-
8	8	8	7	7	7	6	6	-
9	9	9	8	8	7	7	7	7

Tabela 5.5: Diâmetros de Ordenação de λ-Permutações com Sinais por λ-Reversões e λ-Transposições.

5.6.1 Diâmetro de Ordenação de λ-Permutações sem Sinais por λ-Reversões

Para o problema de Ordenação de λ-Permutações sem Sinais por λ-Reversões, consideraremos a família de λ-permutações π^{r1}, que atinge o diâmetro para $3 \leq n \leq 10$ e $\lambda = 2$. As λ-permutações de tal família serão dadas pela função $\pi^{r1}(n)$, que é definida a seguir.

Se n é par, então:

$$\pi^{r1}(n) = (2\ 1\ 4\ 3\ \ldots\ n\ n-1).$$

Se n é ímpar, então:

$$\pi^{r1}(n) = (2\ 1\ 4\ 3\ \ldots\ n-1\ n-2\ n).$$

O Teorema 16 mostra um algoritmo que utiliza $\lfloor n/2 \rfloor$ 2-reversões para ordenar uma 2-permutação presente em π^{r1}.

Teorema 16. *Existe um algoritmo que ordena uma λ-permutação da família π^{r1} aplicando $\lfloor n/2 \rfloor$ λ-reversões.*

Demonstração. Seja $\pi = \pi^{r1}(n)$ para algum n e seja $\lambda \geq 2$.

Se n é par, então a sequência de ordenação que o algoritmo deve aplicar é composta pelas 2-reversões $\rho(i, i+1)$, para todo $i \in \{1, 3, \ldots, n-1\}$. Caso contrário, tal sequência é composta pelas 2-reversões $\rho(i, i+1)$, para todo $i \in \{1, 3, \ldots, n-2\}$. Assim, o número de operações da sequência é igual $\lfloor n/2 \rfloor$, ou seja, a quantidade de números inteiros ímpares entre 1 e $n-1$. □

Agora, considerando $3 \leq \lambda \leq n \leq 10$, temos exatamente duas famílias de λ-permutações atingindo o diâmetro, denotadas por π^{r2} e π^{r3}. As λ-permutações de tais famílias serão dadas pelas funções $\pi^{r2}(n)$ e $\pi^{r3}(n)$, que são definidas a seguir.

Se n é par, então:

1. $\pi = \pi^{r2}(n)$ é tal que $\pi_1 = 2$, $\pi_n = n-1$, e $\pi_i = i+2$, para todo $2 \leq i \leq n-2$ em que i é par, ou $\pi_i = i-2$, para todo $3 \leq i \leq n-1$ em que i é ímpar;

2. $\pi = \pi^{r3}(n)$ é tal que $\pi_2 = 1$, $\pi_{n-1} = n$, e $\pi_i = i-2$, para todo $4 \leq i \leq n$ em que i é par, ou $\pi_i = i+2$, para todo $1 \leq i \leq n-3$ em que i é ímpar;

Se n é ímpar, então:

1. $\pi = \pi^{r2}(n)$ é tal que $\pi_1 = 2$, $\pi_{n-1} = n$, e $\pi_i = i+2$, para todo $2 \leq i \leq n-3$ em que i é par, ou $\pi_i = i-2$, para todo $3 \leq i \leq n$ em que i é ímpar;

2. $\pi = \pi^{r3}(n)$ é tal que $\pi_2 = 1$, $\pi_n = n-1$, e $\pi_i = i-2$, para todo $4 \leq i \leq n-1$ em que i é par, ou $\pi_i = i+2$, para todo $1 \leq i \leq n-2$ em que i é ímpar;

Por exemplo, $\pi^{r2}(6) = (2\ 4\ 1\ 6\ 3\ 5)$, $\pi^{r2}(7) = (2\ 4\ 1\ 6\ 3\ 7\ 5)$, $\pi^{r3}(6) = (3\ 1\ 5\ 2\ 6\ 4)$ e $\pi^{r3}(7) = (3\ 1\ 5\ 2\ 7\ 4\ 6)$. Os lemas 29 e 30 mostram que uma 2-reversão ou 3-reversão aplicada sobre $\pi^{r2}(n)$ ou $\pi^{r3}(n)$ resulta em uma λ-permutação, respectivamente. Eles são auxiliares ao Teorema 17, que mostra um algoritmo que utiliza $n-1$ operações para ordenar tais famílias de permutações.

Lema 29. *Seja π uma λ-permutação e π_{i+1} o menor elemento tal que $\pi_{i+1} \neq i+1$. Então, $\pi' = \pi \cdot \rho(i, i+1)$ é uma λ-permutação.*

Demonstração. Note que $\pi_{i+1} = i > \pi_i$, $\pi'_i - i = 0 < \lambda$ e π'_{i+1} está mais próximo de sua posição correta do que π_i está em π, logo como π é uma λ-permutação, temos que π' também é uma λ-permutação. □

Lema 30. *Seja π uma λ-permutação e π_{i+2} o menor elemento tal que $\pi_{i+2} \neq i+2$. Então, $\pi' = \pi \cdot \rho(i, i+2)$ é uma λ-permutação.*

Demonstração. Note que $\pi'_{i+1} = \pi_{i+1}$ e $\pi'_i = \pi_{i+2} = i$ e então a prova segue observando o elemento π_i. Temos $i+1 \leq \pi_i = \pi'_{i+2} \leq i+\lambda-1$, pois $\pi_i > \pi_{i+2} = i$ e π é uma λ-permutação, o que implica que $|\pi'_{i+2} - (i+2)| < \lambda$, e logo π' é uma λ-permutação. □

Teorema 17. *Existe um algoritmo que ordena uma λ-permutação da família π^{r2} ou da família π^{r3} aplicando $n-1$ λ-reversões.*

Demonstração. Seja $\pi = \pi^{r2}(n)$ ou $\pi = \pi^{r3}(n)$ para algum n e seja $\lambda \geq 3$ um inteiro.

Considere um algoritmo que repetidamente toma o menor elemento fora de lugar e o move para sua posição correta aplicando uma 2-reversão ou uma 3-reversão. Note que isto sempre é possível já que $|\pi_i - i| < 3$. Além disso, cada permutação resultante gerada no processo será uma λ-permutação, como mostra o Lema 30.

Uma vez que apenas os $n-1$ menores elementos de π precisam ser corretamente posicionados (pois, posterior a isto, o n-ésimo elemento estará na posição n), tal algoritmo utilizará um total de $n-1$ λ-reversões. □

Como mostra a Tabela 5.1, para $3 \leq n \leq 10$, temos $D_r^{\lambda*}(n) = \lfloor n/2 \rfloor$ e $D_r^{\lambda*}(n) = n-1$, para $\lambda = 2$ e $3 \leq \lambda \leq n$, respectivamente. Uma vez que os teoremas 16 e 17 mostram como ordenar λ-permutações que atingem o diâmetro para tais valores de n e λ com o número ótimo de operações, apresentamos a Conjectura 1.

Conjectura 1. *$D_r^{2*}(n) = \lfloor n/2 \rfloor$ e $D_r^{\lambda*}(n) = n-1$, para quaisquer n e λ tais que $n \geq 3$ e $3 \leq \lambda \leq n$.*

5.6.2 Diâmetro de Ordenação de λ-Permutações com Sinais por λ-Reversões

A família de λ-permutações π^{r1}, descrita na Seção 5.6.1, também será considerada para o problema de Ordenação de λ-Permutações com Sinais por λ-Reversões, entretanto para todo $\lambda \geq 2$. Neste problema, λ-permutações de tal família atingem o diâmetro para todo $2 \leq \lambda \leq n \leq 9$ tais que n é par. Além disso, baseado na observação que temos $d_{\bar{r}}^{\lambda*}(\pi^{r1}(1)) = 0$, $d_{\bar{r}}^{\lambda*}(\pi^{r1}(2)) = 3$, $d_{\bar{r}}^{\lambda*}(\pi^{r1}(3)) = 3$ e $d_{\bar{r}}^{\lambda*}(\pi^{r1}(4)) = 5$, o Teorema 18 mostra um algoritmo que ordena uma permutação da família π^{r1} com no máximo $5n/4$ λ-reversões.

Teorema 18. *Existe um algoritmo que ordena uma λ-permutação com sinais da família π^{r1} aplicando no máximo $5\lceil n/4 \rceil$ λ-reversões.*

Demonstração. Seja $\pi = \pi^{r1}(n)$ para algum n e seja $\lambda \geq 2$ um inteiro.

Primeiramente, mapearemos os elementos de π em λ-permutações $\pi^0, \pi^1, \ldots, \pi^{\lceil n/4 \rceil - 1}$, tais que $\pi^i = (\pi_{4i+1} - 4i \ldots \pi_{\min(n, \pi_{4i+4}-4i)})$ para todo $0 \leq i \leq \lceil n/4 \rceil - 1$. Assim, note que uma mesma sequência que ordena uma λ-permutação π^i também é válida para ordenar os elementos mapeados por ela em π. Um algoritmo que ordena cada λ-permutação π^i individualmente utiliza no máximo $5\lceil n/4 \rceil$ λ-operações, pois π^i tem no máximo 4 elementos e $d_{\bar{r}}^{\lambda*}(\pi^{r1}(4)) = 5$. □

O algoritmo descrito no Teorema 18 dá um limitante superior para ordenar uma λ-permutation da família π^{r1}. Como a mesma aparece no diâmetro para os valores de n apresentados na Tabela 5.4 que são pares e, como $D_{\bar{r}}^{\lambda*}(n) \leq D_{\bar{r}}^{\lambda*}(n+1)$ para qualquer $n \geq 1$, concluímos a Conjectura 2.

Conjectura 2. *Para todo n par e $2 \leq \lambda \leq n$, temos $D_{\bar{r}}^{\lambda*}(n) \leq 5\lceil n/4 \rceil$. Para todo n ímpar e $2 \leq \lambda \leq n$, temos $D_{\bar{r}}^{\lambda*}(n) \leq 5\lceil (n+1)/4 \rceil$.*

5.6.3 Diâmetros de Ordenação de λ-Permutações por λ-Transposições, e de λ-Permutações com e sem Sinais por λ-Reversões e λ-Transposições

O Lema 31 mostra que sempre é possível colocar o menor elemento fora de lugar de uma λ-permutação π em sua posição correta aplicando uma λ-transposição. Ele é auxiliar ao Teorema 19, que mostra como ordenar uma λ-permutação sem sinais com $n-1$ λ-transposições. Tal teorema implica no Corolário 13, que mostra uma quantidade de λ-transposições e λ-reversões suficiente para ordenar uma λ-permutação com sinais.

Lema 31. *Seja π uma λ-permutação e $|\pi_j| = i$ o menor elemento fora de lugar em π. Então, $\pi' = \pi \cdot \tau(i, j, j+1)$ é uma λ-permutação e $\tau(i, j, j+1)$ é uma λ-transposição.*

Demonstração. É fácil ver que $\tau(i, j, j+1)$ é uma λ-transposição, pois $|i-j| < \lambda$, já que π é uma λ-permutação. Note que em π', o elemento $|\pi_j| = i$ está na posição i e os elementos $\pi_i, \ldots, \pi_{j-1}$ foram movidos exatamente uma posição para a direita. Como $|\pi_j| = i$ é o menor elemento fora de lugar em π, temos $\pi_i, \ldots, \pi_{j-1}$ mais próximos de suas posições corretas em π' e logo, π' λ-permutação. □

Teorema 19. *Existe um algoritmo que ordena uma λ-permutação sem sinais com no máximo $n-1$ λ-transposições.*

Demonstração. Seja π uma λ-permutação e seja $\lambda \geq 2$ um inteiro.

Considere um algoritmo que repetidamente toma o menor elemento fora de lugar e o move para sua posição correta aplicando uma λ-transposição. Note que isto sempre é possível e também que a permutação resultante é uma λ-permutação, como mostra o Lema 31.

Uma vez que apenas os $n-1$ menores elementos de π precisam ser corretamente posicionados (pois, posterior a isto, o n-ésimo elemento estará na posição n), tal algoritmo utilizará um total de $n-1$ λ-transposições. □

Corolário 13. *Existe um algoritmo que ordena uma λ-permutação com sinais com no máximo $n-1$ λ-transposições e n λ-reversões unitárias.*

Note que o Teorema 19 é válido para ordenar λ-permutações sem sinais considerando modelos de rearranjo que permitem apenas λ-transposições, ou ambas as λ-operações. E, adicionalmente ao Corolário 13, concluímos o Teorema 20.

Teorema 20. $D_t^{\lambda*}(n) \leq n-1$, $D_{rt}^{\lambda*}(n) \leq n-1$ *e* $D_{\bar{r}t}^{\lambda*}(n) \leq 2n-1$.

5.7 Ordenação de Permutações por λ-Reversões

Como dito anteriormente, o objetivo do problema de Ordenação de Permutações por λ-Reversões é transformar uma permutação na permutação identidade aplicando o menor número de λ-reversões possível. Esta seção apresenta algoritmos de 10-aproximação, baseados na definição de *breakpoints*, para λ-permutações com e sem sinais dadas como entrada em tal problema, e note então que não há garantia de que as permutações intermediárias também serão λ-permutações, pois esta não é uma restrição do problema em questão. Além disso, estes são os primeiros algoritmos com fatores de aproximação constantes que apresentamos para tal problema.

O Lema 32 apresenta como obter uma permutação $\pi \cdot \tau(i, j, k+1)$ com pelo menos um *breakpoint* a menos ao aplicar 4 λ-reversões sobre uma λ-permutação. Então, o Teorema 21 mostra algoritmos de 10-aproximação para os problemas de Ordenação de Permutações com e sem Sinais por λ-Reversões, considerando λ-permutações como entrada.

Lema 32. *Seja π uma λ-permutação. Seja $|\pi_j| = i$ o menor elemento fora de lugar em π. Suponha que π_j é uma strip crescente $S = (\pi_j \dots \pi_k)$. É sempre possível obter uma permutação $\pi \cdot \tau(i, j, k+1)$ com no máximo $b(\pi) - 1$ breakpoints aplicando no máximo 4 λ-reversões.*

Demonstração. Seja $R = (\pi_i \dots \pi_{j-1})$ o segmento que será movido para a direita em $\tau(i, j, k+1)$. Temos $|S| \leq \lambda - 1$ e $|R| \leq \lambda - 1$, como mostra o Lema 2. A ideia é aplicar uma sequência com no máximo 4 λ-reversões que coloca os elementos de S em suas posições ao revertê-los com os elementos de R. Então, seguimos dividindo ambos segmentos R e S em no máximo 2 segmentos cada, onde cada subsegmento tem no máximo $\lfloor \lambda/2 \rfloor$ elementos. Os seguintes casos podem acontecer:

(i) $|S| > \lfloor \lambda/2 \rfloor$ e $|R| > \lfloor \lambda/2 \rfloor$: Sejam $A = (\pi_i \dots \pi_{i+\lfloor \lambda/2 \rfloor - 1})$ e $B = (\pi_{i+\lfloor \lambda/2 \rfloor} \dots \pi_{j-1})$ os subsegmentos de R. Sejam $C = (\pi_j \dots \pi_{j+\lfloor \lambda/2 \rfloor - 1})$ e $D = (\pi_{j+\lfloor \lambda/2 \rfloor} \dots \pi_k)$ os subsegmentos de S. Também, estaremos nos referindo a um subsegmento de acordo com os elementos presentes nele, por exemplo, se revertermos o subsegmento $(\pi_i \dots \pi_{i+\lfloor \lambda/2 \rfloor - 1})$, ele ainda será referido como A. Então, para alcançar $\pi \cdot \tau(i, j, k+1)$, nosso objetivo é trocar a posição do subsegmento A com o subsegmento C, e de B com D, enquanto mantemos a ordem original dos elementos em cada subsegmento. Isto pode ser facilmente feito ao reverter C juntamente com B, C juntamente com A, D juntamente com B e, finalmente, D juntamente com A. Note que cada subsegmento é envolvido em duas reversões e então todos eles terminam com seus elementos na ordem inicial. Já que C termina com seus elementos a partir da posição i e D termina com seus elementos a partir da posição $i + \lfloor \lambda/2 \rfloor$, os elementos de S terminam em suas posições corretas (de acordo com a permutação identidade).

(ii) $|S| \leq \lfloor \lambda/2 \rfloor$ e $|R| \leq \lfloor \lambda/2 \rfloor$: Neste caso, revertemos inicialmente S juntamente com R para colocar os elementos de S a partir da posição i e os elementos de R a partir da posição j. Já que ambos estão com seus elementos revertidos, mais 2 reversões precisam ser aplicadas, totalizando 3 reversões para alcançar $\pi \cdot \tau(i, j, k+1)$.

(iii) $S \leq \lfloor \lambda/2 \rfloor$ e $R > \lfloor \lambda/2 \rfloor$: Seja $A = (\pi_i \dots \pi_{i+\lfloor \lambda/2 \rfloor - 1})$ e $B = (\pi_{i+\lfloor \lambda/2 \rfloor} \dots \pi_{j-1})$ subsegmentos de R. Seja $C = (\pi_j \dots \pi_k)$ o subsegmento de S. Novamente, estaremos

nos referindo a um subsegmento de acordo com os elementos presentes nele. Assim, para alcançar $\pi \cdot \tau(i, j, k+1)$, primeiro colocamos os elementos de C a partir da posição i ao revertê-los com B, e então com A. Já que C foi revertido duas vezes e C tem todos os elementos de S, seus elementos terminam em suas posições corretas (de acordo com ι). Apesar dos elementos de A e B estarem a partir das posições j e $j+\lfloor\lambda/2\rfloor$, respectivamente, seus elementos estão revertidos e então mais 2 reversões devem ser aplicadas sobre ambos subsegmentos, totalizando 4 operações para obter $\pi \cdot \tau(i, j, k+1)$.

(iv) $S > \lfloor\lambda/2\rfloor$ e $R \leq \lfloor\lambda/2\rfloor$: Simétrico ao caso (iii).

Já que todas reversões descritas envolvem no máximo 2 subsegmentos de tamanho no máximo $\lfloor\lambda/2\rfloor$, elas também são λ-reversões e, como mostrado no Lema 24, a permutação obtida $\pi \cdot \tau(i, j, k+1)$ tem pelo menos um *breakpoint* a menos que π. □

Apesar da permutação $\pi \cdot \tau(i, j, k+1)$ ser uma λ-permutação, note que o Lema 32 não garante que todas as permutações geradas no processo para alcançá-la são λ-permutações. Por exemplo, considere uma 7-permutação sem sinais $\pi = (2\ 3\ 4\ 5\ 6\ 12\ 1\ 7\ 8\ 9\ 10\ 11)$, então $\pi_j = i = 1$ e $S = (1)$. Ao seguir a prova, temos $R = (2\ 3\ 4\ 5\ 6\ 12)$ e, indo ao caso (iii), temos também $A = (2\ 3\ 4)$ e $B = (5\ 6\ 12)$ como subsegmentos de R, e $C = (1)$ como subsegmento de S. Depois de reverter C com B, obtemos uma permutação $\pi = (2\ 3\ 4\ 1\ 12\ 6\ 5\ 7\ 8\ 9\ 10\ 11)$, que não é uma 7-permutação, visto que o elemento 12 foi movido para a posição 5.

Teorema 21. *Os problemas de Ordenação de Permutações com e sem Sinais por λ-Operações têm algoritmos* 10*-aproximação para λ-permutações dadas como entrada.*

Demonstração. Seja $\lambda \geq 2$ um inteiro e seja $\pi \neq \iota$ uma λ-permutação.

Considere um algoritmo que primeiro aplica uma λ-reversão sobre cada *strip* decrescente em π para obter uma permutação que contenha apenas *strips* crescentes. Posteriormente, ele repetidamente toma o menor elemento fora de lugar e move a *strip* que o contém para sua posição correta.

Como mostrado no Lema 32, no máximo 4 λ-reversões são necessárias para mover cada *strip* para sua posição correta, e já que as operações simulam uma transposição, não haverá novas *strips* decrescentes na permutação resultado.

Já que, talvez, uma λ-reversão pôde ter sido aplicada sobre cada *strip* decrescente no primeiro passo do algoritmo para torná-la crescente, temos que no máximo 5 λ-reversões são necessárias para remover pelo menos um *breakpoint*. Logo, o número de operações do nosso algoritmo é no máximo $5b(\pi) \leq 10d_\beta^\lambda(\pi)$, para $\beta \in \{r, \bar{r}\}$, e a inequação segue do Lema 1. □

Capítulo 6

Conclusão

Nesta dissertação nós introduzimos o estudo de dois problemas: (i) Ordenação de Permutações por λ-Operações e (ii) Ordenação de λ-Permutações por λ-Operações. Em ambos os problemas, consideramos permutações com e sem sinais, e modelos de rearranjo permitindo apenas λ-reversões, apenas λ-transposições e também ambas as operações, caracterizando 5 variações de cada um dos problemas.

Para (i), apresentamos 15 algoritmos de aproximação, sendo 3 algoritmos para cada variação considerada, onde um deles tem melhor fator para valores grandes de λ e os outros dois tem melhor fator para valores pequenos de λ. Além disso, enfatizamos que os fatores de aproximação de todos os algoritmos apresentados para este problema dependem de λ e/ou do tamanho da permutação.

Para (ii), mostramos limitantes inferiores e superiores para o número de λ-permutações existentes e, além disso, desenvolvemos algoritmos com fatores de aproximação de $O(\lambda^2)$, $O(\lambda)$, e $O(1)$ para as variações estudadas. Ademais, foram apresentados resultados sobre os valores dos diâmetros de ordenação de cada problema de ordenação de λ-permutações abordados. Finalmente, também foram mostrados algoritmos com fatores de aproximação $O(1)$ para qualquer λ-permutação dada como entrada nos problemas de Ordenação de Permutações com e sem Sinais por λ-Reversões.

Para ambos os problemas foram apresentados resultados experimentais que comparam os fatores de aproximação médios e/ou máximos dos algoritmos desenvolvidos numa perspectiva prática.

Alguns dos resultados para os problemas de Ordenação de Permutações sem Sinais por λ-Operações foram publicados em 2018 na conferência AlCoB *(International Conference on Algorithms for Computational Biology)* [41]. Posteriormente, também em 2018, alguns dos resultados para os problemas de Ordenação de λ-Permutações sem Sinais por λ-Operações foram publicados na conferência BSB (*Brazilian Symposium on Bioinformatics*) [40].

Como trabalhos futuros, além de investigar algoritmos com fatores de aproximação melhores para (i) e (ii), sugerimos os estudos de variações de ambos os problemas que consideram λ-operações ponderadas e/ou as regiões intergênicas dos genomas. Ambas as variações tendem a adicionar relevância biológica a esses problemas [1, 8, 9].

Referências Bibliográficas

[1] Alexsandro Oliveira Alexandrino, Carla Negri Lintzmayer, and Zanoni Dias. Approximation Algorithms for Sorting Permutations by Fragmentation-Weighted Operations. In *Algorithms for Computational Biology*, volume 10849, pages 53–64. Springer International Publishing, Heidelberg, Germany, 2018.

[2] David A. Bader, Bernard M. E. Moret, and Mi Yan. A Linear-Time Algorithm for Computing Inversion Distance Between Signed Permutations with an Experimental Study. *Journal of Computational Biology*, 8:483–491, 2001.

[3] Vineet Bafna and Pavel A. Pevzner. Sorting by Reversals: Genome Rearrangements in Plant Organelles and Evolutionary History of X Chromosome. *Molecular Biology and Evolution*, 12(2):239–246, 1995.

[4] Vineet Bafna and Pavel A. Pevzner. Genome Rearrangements and Sorting by Reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[5] Vineet Bafna and Pavel A. Pevzner. Sorting by Transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, 1998.

[6] Anne Bergeron. A Very Elementary Presentation of the Hannenhalli-Pevzner Theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.

[7] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-Approximation Algorithm for Sorting by Reversals. In R. Möhring and R. Raman, editors, *Proceedings of the 10th Annual European Symposium on Algorithms (ESA'2002)*, volume 2461 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag Berlin Heidelberg New York, Berlin/Heidelberg, Germany, 2002.

[8] Priscila Biller, Laurent Guéguen, Carole Knibbe, and Eric Tannier. Breaking Good: Accounting for Fragility of Genomic Regions in Rearrangement Distance Estimation. *Genome Biology and Evolution*, 8(5):1427–1439, 2016.

[9] Priscila Biller, Carole Knibbe, Guillaume Beslon, and Eric Tannier. Comparative Genomics on Artificial Life. In *Pursuit of the Universal*, pages 35–44. Springer International Publishing, 2016.

[10] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Sorting by Transpositions is Difficult. *SIAM Journal on Computing*, 26(3):1148–1180, 2012.

[11] Laurent Bulteau, Guillaume Fertin, and Irena Rusu. Pancake Flipping is Hard. *Journal of Computer and System Sciences*, 81(8):1556–1574, 2015.

[12] Alberto Caprara. Sorting Permutations by Reversals and Eulerian Cycle Decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110, 1999.

[13] Timothy M Chan and Mihai Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 161–173. Society for Industrial and Applied Mathematics, 2010.

[14] Xin Chen. On Sorting Unsigned Permutations by Double-Cut-and-Joins. *Journal of Combinatorial Optimization*, 25(3):339–351, 2013.

[15] David A. Christie. A 3/2-Approximation Algorithm for Sorting by Reversals. In H. Karloff, editor, *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'1998)*, pages 244–252, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[16] David A. Christie. *Genome Rearrangement Problems*. PhD thesis, Department of Computing Science, University of Glasgow, 1998.

[17] Daniel A. Dalevi, Niklas Eriksen, Kimmo Eriksson, and Siv G. E. Andersson. Measuring Genome Divergence in Bacteria: A Case Study Using Chlamydian Data. *Journal of Molecular Evolution*, 55(1):24–36, 2002.

[18] Zanoni Dias and João Meidanis. Sorting by Prefix Transpositions. In A. H. F. Laender and A. L. Oliveira, editors, *Proceedings of the 9th International Symposium on String Processing and Information Retrieval (SPIRE'2002)*, volume 2476 of *Lecture Notes in Computer Science*, pages 65–76. Springer-Verlag Berlin Heidelberg New York, Berlin/Heidelberg, Germany, 2002.

[19] Harry Dweighter. Problem E2569. *American Mathematical Monthly*, 82:1010, 1975.

[20] Isaac Elias and Tzvika Hartman. A 1.375-Approximation Algorithm for Sorting by Transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):369–379, 2006.

[21] Johannes Fischer and Simon W. Ginzinger. A 2-Approximation Algorithm for Sorting by Prefix Reversals. In G. S. Brodal and S. Leonardi, editors, *Proceedings of the 13th Annual European Conference on Algorithms (ESA'2005)*, volume 3669 of *Lecture Notes in Computer Science*, pages 415–425, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[22] Gustavo R. Galvão, Christian Baudet, and Zanoni Dias. Sorting Signed Circular Permutations by Super Short Reversals. In R. Harrison, Y. Li, and I. Măndoiu, editors, *Proceedings of the 11th International Symposium on Bioinformatics Research and Applications (ISBRA'2015)*, Lecture Notes in Computer Science, pages 272–283. Springer International Publishing, Switzerland, 2015.

[23] Gustavo R. Galvão, Orlando Lee, and Zanoni Dias. Sorting Signed Permutations by Short Operations. *Algorithms for Molecular Biology*, 10(1):1–17, 2015.

[24] Gustavo Rodrigues Galvão and Zanoni Dias. Approximation Algorithms for Sorting by Signed Short Reversals. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB'2014)*, pages 360–369, New York, NY, USA, 2014. ACM.

[25] Gustavo R. Galvão, Christian Baudet, and Zanoni Dias. Sorting Circular Permutations by Super Short Reversals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(3):620–633, 2017.

[26] William H. Gates and Christos H. Papadimitriou. Bounds for Sorting by Prefix Reversal. *Discrete Mathematics*, 27(1):47–57, 1979.

[27] Qian-Ping Gu, Shietung Peng, and Ivan H. Sudborough. A 2-Approximation Algorithm for Genome Rearrangements by Reversals and Transpositions. *Theoretical Computer Science*, 210(2):327–339, 1999.

[28] Sridhar Hannenhalli and Pavel A. Pevzner. Transforming Cabbage into Turnip: Polynomial Algorithm for Sorting Signed Permutations by Reversals. *Journal of the ACM*, 46(1):1–27, 1999.

[29] Lenwood S. Heath and John P. C. Vergara. Sorting by Bounded Block-moves. *Discrete Applied Mathematics*, 88(1-3):181–206, 1998.

[30] Lenwood S. Heath and John Paul C. Vergara. Sorting by Short Block-Moves. *Algorithmica*, 28(3):323–352, 2000.

[31] Lenwood S. Heath and John Paul C. Vergara. Sorting by Short Swaps. *Journal of Computational Biology*, 10(5):775–789, 2003.

[32] Mark R. Jerrum. The Complexity of Finding Minimum-length Generator Sequences. *Theoretical Computer Science*, 36(2-3):265–289, 1985.

[33] Haitao Jiang, Haodi Feng, and Daming Zhu. An 5/4-Approximation Algorithm for Sorting Permutations by Short Block Moves. In H. Ahn and C. Shin, editors, *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC'2014)*, volume 8889 of *Lecture Notes in Computer Science*, pages 491–503. Springer International Publishing, 2014.

[34] Haitao Jiang, Daming Zhu, and Binhai Zhu. A (1+e)-Approximation Algorithm for Sorting by Short Block-Moves. *Theoretical Computer Science*, 437:1–8, 2012.

[35] John D. Kececioglu and David Sankoff. Exact and Approximation Algorithms for Sorting by Reversals, with Application to Genome Rearrangement. *Algorithmica*, 13:180–210, 1995.

[36] Donald E. Knoth. *Fundamental Algorithms: The art of Computer Programming*. 1973.

[37] Jean-François Lefebvre, Nadia El-Mabrouk, Elisabeth R. M. Tillier, and David Sankoff. Detection and validation of single gene inversions. *Bioinformatics*, 19(1):i190–i196, 2003.

[38] Guohui Lin and Tao Jiang. A Further Improved Approximation Algorithm for Breakpoint Graph Decomposition. *Journal of Combinatorial Optimization*, 8(2):183–194, 2004.

[39] Carla Negri Lintzmayer, Guillaume Fertin, and Zanoni Dias. Sorting Permutations by Prefix and Suffix Rearrangements. *Journal of Bioinformatics and Computational Biology*, 15(1):1750002, 2017.

[40] Guilherme Henrique Santos Miranda, Alexsandro Oliveira Alexandrino, Carla Negri Lintzmayer, and Zanoni Dias. Sorting λ-Permutations by λ-Operations. In *Proceedings of the 11th Brazilian Symposium on Bioinformatics (BSB'2018)*, pages 1–13. Springer International Publishing, Heidelberg, Germany, 2018.

[41] Guilherme Henrique Santos Miranda, Carla Negri Lintzmayer, and Zanoni Dias. Sorting Permutations by Limited-Size Operations. In *Algorithms for Computational Biology*, volume 10849, pages 76–87. Springer International Publishing, Heidelberg, Germany, 2018.

[42] Pavel A. Pevzner and Michael S. Waterman. Open Combinatorial Problems in Computational Molecular Biology. In *Proceedings of the 3rd Israel Symposium on the Theory of Computing Systems (ISTCS'1995)*, pages 158–173, Washington, DC, USA, 1995. IEEE Computer Society.

[43] Atif Rahman, Swakkhar Shatabda, and Masud Hasan. An Approximation Algorithm for Sorting by Reversals and Transpositions. *Journal of Discrete Algorithms*, 6(3):449–457, 2008.

[44] Irena Rusu. Log-Lists and Their Applications to Sorting by Transpositions, Reversals and Block-Interchanges. *Theoretical Computer Science*, 660:1–15, 2017.

[45] Mahfuza Sharmin, Rukhsana Yeasmin, Masud Hasan, Atif Rahman, and Mohammad S. Rahman. Pancake Flipping with Two Spatulas. *Electronic Notes in Discrete Mathematics*, 36:231–238, 2010.

[46] Eric Tannier, Anne Bergeron, and Marie-France Sagot. Advances on Sorting by Reversals. *Discrete Applied Mathematics*, 155(6-7):881–888, 2007.

[47] Glenn Tesler. GRIMM: Genome Rearrangements Web Server. *Bioinformatics*, 18(3):492–493, 2002.

[48] John P. C. Vergara. *Sorting by Bounded Permutations*. PhD thesis, Virginia Polytechnic Institute and State University, 1998.

[49] Maria E. M. T. Walter, Zanoni Dias, and João Meidanis. Reversal and Transposition Distance of Linear Chromosomes. In *Proceedings of the 5th International Symposium on String Processing and Information Retrieval (SPIRE'1998)*, pages 96–102, Los Alamitos, CA, USA, 1998. IEEE Computer Society.

[50] Maria E. M. T. Walter, Zanoni Dias, and João Meidanis. A New Approach for Approximating The Transposition Distance. In F. M. Titsworth, editor, *Proceedings of the 7th String Processing and Information Retrieval (SPIRE'2000)*, pages 199–208, Los Alamitos, CA, USA, 2000. IEEE Computer Society.